MAGAZINE BSD

FOR NOVICE AND ADVANCED USERS

FreeBSD 10.2

USING THE FREEBSD'S PROCSTAT API IN A WEB CONTEXT

THE BASIC SEMANTICS OF *UNIX UNITED*

IS TRUENAS JUST FREENAS INSTALLED ON A SERVER?

HOW ABOUT SOME RASPBERRY PI?

VOL.9 NO.08
ISSUE 72
1898-9144

FREENAS MINI
STORAGE APPLIANCE

IT *SAVES* YOUR LIFE.

HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**

Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time*.**

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured

iXsystems®

http://www.iXsystems.com/mini

intel® inside™ ATOM™

FREENAS CERTIFIED STORAGE

With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

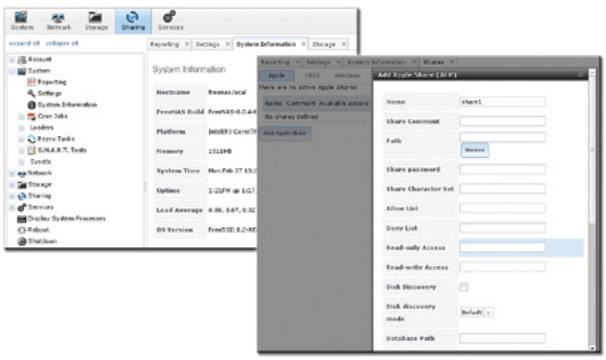
MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

» Custom built and optimized for your use case
» Installed, configured, tested, and guaranteed to work out of the box
» Supported by the Silicon Valley team that designed and built it
» Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

http://www.iXsystems.com/storage/freenas-certified-storage/

Dear Readers,

We would like to introduce a new issue made by the BSD Team. This time you will deal with Unix and FreeBSD topics. You will learn more about the basic semantics of Unix United and you will learn how to start terminal in Unix. Reading our step-by-step tutorials will give you professional expertise in the subjects presented. You will get to know how to use the FreeBSD's procstat API in a web context. David will present this topic to you in his article which can be found on page 14.

Most of you are very familiar with FreeNAS, and I would like to invite you to read the Expert Says column to check out "What's the Difference Between TrueNAS and FreeNAS? Is TrueNAS Just FreeNAS Installed on a Server?" All your questions will be covered in this article written by Brett Davis.

In this issue we also continue to write about the Raspberry Pi and I hope those of you who need and want to expand the knowledge on this topic will find the article by Jerry Craft very useful and interesting.

Finally, please do not forget to see the next column by Rob Somerville. This time you will also find two new materials prepared especially for you. We decided to start publishing the monthly news from the BSD world for you. We selected the best news about products, OSes and events from the last month and we announced the upcoming conferences to keep you up-to-date. The second one is a Quiz prepared by Rob Somerville. You will find the quiz next to the News column and the answers are published on the last page of the issue. This way you can test your knowledge. Have great fun!

As always, I would like to thank you all for really great articles and your willingness to help me create this issue of BSD magazine.

Enjoy reading!
Ewa and BSD Team

MAGAZINE BSD

Editor in Chief:
Ewa Dudzic
ewa.dudzic@software.com.pl

Contributing:
Michael Shirk, Andrey Vedikhin, Petr Topiarz, Solène Rapenne, Anton Borisov, Jeroen van Nieuwenhuizen, José B. Alós, Luke Marsden, Salih Khan, Arkadiusz Majewski, BEng, Toki Winter, Wesley Mouedine Assaby, Rob Somerville

Top Betatesters & Proofreaders:
Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth, Ben Milman, Mark VonFange

Special Thanks:
Annie Zhang
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher:
Paweł Marciniak
pawel@software.com.pl

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

Publisher:
Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

FreeNAS

in an Enterprise Environment

By the time you're reading this, FreeNAS has been downloaded more than 5.5 million times. For home users, it's become an indispensable part of their daily lives, akin to the DVR. Meanwhile, all over the world, thousands of businesses universities, and government departments use FreeNAS to build effective storage solutions in myriad applications.

What you will learn...

- How TrueNAS builds off the strong points of the FreeBSD and FreeNAS operating systems
- How TrueNAS meets modern storage challenges for enterr

The FreeNAS operating systems is fre
the public and offers thorough doc
active community, and a feature-ric
the storage environment. Based on Free
can share over a host of protocols (SMB
FTP, iSCSI, etc) and features an intuitiv
the ZFS file system, a plug-in system
much more.

Despite the massive popularity
aren't aware of its big brother dut
data in some of the most demand
environments: the proven, enterp
professionally-supported line of

But what makes TrueNAS diffe
Well, I'm glad you asked...

Commercial Grade Suppo

When a mission critical stor
organization's whole operat
halt. Whole community-bas
free), it can't always get an
and running in a timely m
responsiveness and expe
dedicated support team
provide that safety.

Created by the sam
developed FreeNAS.

CONTENTS

Attend

InterDrone

The International Drone Conference and Exposition

InterDrone is Three Awesome Conferences:

Drone TECHCON

For Builders

More than 35 classes, tutorials and panels for hardware and embedded engineers, designers and software developers building commercial drones and the software that controls them.

Drone FLYER

For Flyers and Buyers

More than 35 tutorials and classes on drone operations, flying tips and tricks, range, navigation, payloads, stability, avoiding crashes, power, environmental considerations, which drone is for you, and more!

Drone BUSINESS

For Business Owners, Entrepreneurs & Dealers

Classes will focus on running a drone business, the latest FAA requirements and restrictions, supporting and educating drone buyers, marketing drone services, and where the next hot opportunities are likely to be!

The Largest Commercial Drone Show in North America

Meet with 80+ exhibitors!
Demos! Panels! Keynotes!
The Zipline!

September 9-10-11, 2015
Rio, Las Vegas
www.InterDrone.com

A BZ Media Event

FreeBSD 10.2 Released

The FreeBSD Team announced that the FreeBSD 10.2 is available now. This is the stable version which improves on the stability of FreeBSD 10.1-RELEASE and has the new features. The most relevant features are:

- The resolvconf(8) utility has been updated to version 3.7.0, with improvements to protect DNS privacy.
- The ntp suite has been updated to version 4.2.8p3.
- A new rc(8) script, growfs, has been added, which will resize the root filesystem on boot if the/firstboot file exists.
- The Linux® compatibility version has been updated to support Centos™ 6 ports.
- The drm code has been updated to match Linux® version 3.8.13, allowing running multiple X servers simultaneously.
- Several enhancements and updates for improved FreeBSD/arm support.
- Several ZFS performance and reliability improvements.
- GNOME has been updated to version 3.14.2.
- KDE has been updated to version 4.14.3.
- And much more...

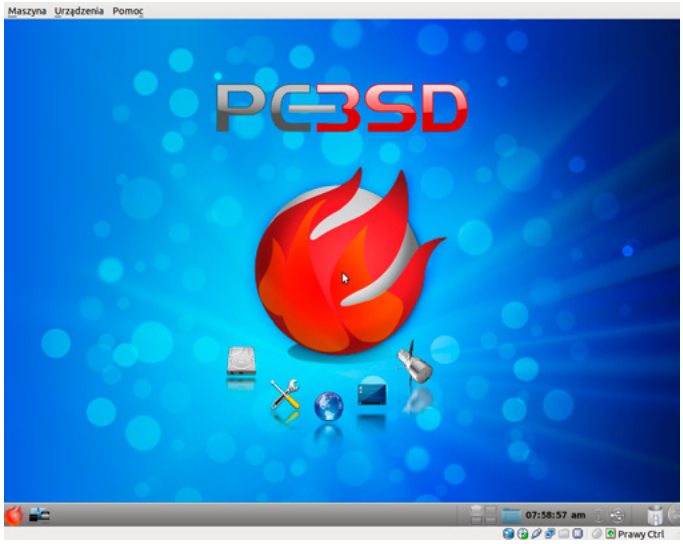
FreeBSD 10.2-RELEASE is available now for the amd64, i386, ia64, powerpc, powerpc64, sparc64, and armv6 architectures and it can be installed from bootable ISO images, or it can be installed from a USB memory stick. The required files can be downloaded via FTP.

https://www.freebsd.org/releases/10.2R/announce.html

PC-BSD 10.2-RC1 Released

The PC-BSD team announced that RC1 images for the upcoming 10.2 release is available now to download. The new improvements of PC-BSD 10.2 are

- FreeBSD 10.2 base system
- Many bugfixes and enhancements to installer to dual-boot setups
- New CD-sized network installation media, with WiFi Configuration via GUI
- Switched to “iocage” for jail management backend
- Disk Manager GUI now available via installer GUI
- Bug-fixes and improvements to Life-Preserver replications
- Improved localization options for login manager
- Options to Enable / Disable SSHD or IPv6 at installation
- New “Plugins” system for AppCafe, allowing download of pre-built jail environments



- Improvements to look-n-feel of AppCafe for package management
- Improved fonts and better support for 4K monitor set-ups
- Enterprise package repo, which only has security updates, allowing users to run a server / desktop or jail with fairly consistent package versions.
- FireFox 39.0
- Chromium 43.0.2357.134
- Thunderbird 38.1.0
- Lumina 0.8.6

10.2-RC1 DVD/USB media can be downloaded from the following URL via HTTP or Torrent.

http://download.pcbsd.org/iso/10.2-RELEASE/edge/amd64/

The New "FreeBSD Mastery: ZFS" Book

ZFS, the fast, flexible, self-healing file system, revolutionized data storage. Leveraging ZFS changes everything about managing FreeBSD systems. With FreeBSD Mastery: ZFS, you'll learn to:

- understand how your hardware affects ZFS
- arrange your storage for optimal performance
- configure datasets that match your enterprise's needs
- repair and monitor storage pools
- expand your storage
- use compression to enhance performance
- determine if deduplication is right for your data
- understand how copy-on-write changes everything
- snapshot file systems
- automatically rotate snapshots
- clone file systems
- understand how ZFS uses and manages space
- do custom FreeBSD ZFS installs

Whether you're a long-term FreeBSD administrator or a new user, FreeBSD Mastery: ZFS will help you simplify storage.

https://www.michaelwlucas.com/nonfiction/freebsd-mastery-zfs
https://www.freebsdmall.com/cgi-bin/fm/bsdmzfs?id=UGkGF4io&mv_pc=194

vBSDcon 2015: September 11-13

VBSDcon will be held on September 11-13, 2015 at the Sheraton in Reston, Virginia. This event will bring together all the BSD community members for a series of roundtable discussions, educational sessions, best practice conversations, and exclusive networking opportunities. You will meet the speakers: Brian Callahan, Bryce Chidester, Michael Dexter, Allan Jude, George Neville-Neil, Pierre Pronchery, Jim Thompson, Willem Toorop, Chang-Hsien Tsai, Shawn Webb, Christos Zoulas and the topics for this year are:

- Supporting a BSD Project
- FreeBSD Virtualization Options
- Made to Measure: Network Performance Analysis in FreeBSD
- What is EdgeBSD?
- blacklist'd: A NetBSD Project
- getdns, A New Stub Resolver
- Interesting things you didn't know you could do with ZFS
- HardenedBSD Internals
- Improving MemGuard Support for UMA on FreeBSD
- Devio.us, the Free OpenBSD Shell Provider and Online *BSD User Group: Technical and Social Lessons Learned from Half a Decade of Service

www.vBSDcon.com

EuroBSDcon 2015

EuroBSDcon is the premier European conference on the open source BSD operating systems attracting about 250 highly skilled engineering professionals, software developers, computer science students and professors, and users from all over Europe and other parts of the world. The goal of EuroBSDcon is to exchange knowledge about the BSD operating systems, facilitate coordination and cooperation among users and developers.

Tutorials will be held in the main conference hotel on Thursday 1st and Friday 2nd of October. The EuroBSD Conference will be on Saturday 3rd and Sunday 4th of October at Stockholm University. You will be able to take the BSD Administration Certificate Exam at the EuroBSDCon 2015. Dru Lavigne has offered to run examinations for people wanting to take the exam. You can see now the list of accepted talks.

Brandon Mercer	Why OpenBSD matters in the Healthcare Industry
Vadim Zhukov	Raceless network configuration
Henning Brauer	OpenBSD sucks
Tommi Pernilä & Arto Jonsson	Attacking FreeBSD network protocols – Why, How and the Results
Ted Unangst	Cryptography in OpenBSD: Another Overview
George Neville-Neil, Jim Thompson	Measure Twice, Code Once
Kirk McKusick	A Brief History of the BSD Fast Filesystem
Andrew Turner	FreeBSD on Arm64
John-Mark Gurney	FreeBSD TLS and crypto performance
Jasper Lievisse Adriaanse	Portroach, OpenBSD distfile scanner
Marc Espie	Faster and more secure packages in OpenBSD
Scott Long	Multiqueue I/O in FreeBSD using LSI and NVME
Ingo Schwarze	Mandoc talk
Stefan Sperling	OpenBSD softraid boot
Mateusz Kocielski	BSD-licensed SASL library
Ed Schouten	CloudABI
Sevan Janiyan	Synchronisation of userland source among BSDs
Jordan Hubbard	Making FreeBSD more dynamic: A year of hacking on asynchronous, centralized interfaces
Masao Uebayashi	config – Rethinking kernel build
François Tigeot	State of the graphics stack in DragonFly
Baptiste Daroussin	Poudrière: efficient package building
Arun Thomas	RISC-V: Berkeley Hardware for Your Berkeley Software (Distribution)
Anders Magnusson	A vacuum-tube computer (that runs BSD)
Taylor R. Campbell	Tricky issues in file systems
Taylor R. Campbell	Protobufs for kernel/user interface

https://2015.eurobsdcon.org/

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES
1.925.240.6652

$39.95
FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

$29.95
PC-BSD 9.1 DVD

$49.95 Save a BUNDLE!
The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

$99.95
The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD

Stylish Dress Attire
Look Your Professional Best

Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD $39.95

CD Set Contains:

- **Disc 1** Installation Boot LiveCD (i386)
- **Disc 2** Essential Packages Xorg (i386)
- **Disc 3** Essential Packages, GNOME2 (i386)
- **Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD $39.95
FreeBSD 9.0 DVD $39.95

FreeBSD Subscriptions

Save time and $$$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 ..... $29.95
FreeBSD Subscription, start with DVD 9.1 ..... $29.95
FreeBSD Subscription, start with CD 9.0 ..... $29.95
FreeBSD Subscription, start with DVD 9.0 ..... $29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD $29.95
PC-BSD Subscription $19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) $39.95
The FreeBSD Handbook, Volume 2 (Admin Guide) $39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) $59.95
The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 ..... $79.95

PC-BSD 9.0 Users Handbook $24.95

BSD Magazine $11.99

The FreeBSD Toolkit DVD $39.95

FreeBSD Mousepad $10.00

FreeBSD & PCBSD Caps $20.00

BSD Daemon Horns $2.00

Bundle Specials!
Save $$$

Just Plain Fun
Mousepads & Novelty Horns

BSD Magazine
Available Monthly

For even **MORE** items
visit our website today!

www.FreeBSDMall.com

The BSD Magazine Summer Quiz

1. What does OWASP stand for?
2. What does an IP flag of 0x40 stand for?
3. Under the USB 3.1 specification is link power management synchronous?
4. Is a netmask of 255.255.255.255 valid?
5. Is 10.2 the latest production release of FreeBSD?
6. When was ZFS incorporated in FreeBSD?
7. Has FreeBSD 10.0-RELEASE reached end of life yet?
8. What is octal 167 in binary?
9. What does BOFH stand for?
10. Are all Android devices vulnerable to a MMS attack that do not require user intervention?
11. What is Bill Gates middle name?
12. Did IBM clear $100,000 million revenue in 2014?
13. What file-system designer was convicted of 2nd degree murder?
14. What is the maximum recommended length of CAT6 cable in an 10GBASE-T electrically noisy environment?
15. If a file starts with 0xFFE or 0XFFF what file type is it likely to be?
16. Is the word volatile a reserved C keyword?
17. What mainframe helped regularise the standard of 8 nits to a byte?
18. What does FIFO stand for?
19. What IBM network protocol supports identical MAC addresses?
20. What does IETF stand for?
21. Where does Theo de Raadt live?
22. What Stanford professor eschewed silicon transistors?
23. What computer was developed by Tommy Flowers?
24. When was PC DOS version 1.0 shipped?
25. What 16 bit microcomputer did BSD originally run on?

Among clouds Performance and Reliability is critical

Download syslog-ng Premium Edition product evaluation here

Attend to a free logging tech webinar here

BalaBit IT Security

www.balabit.com

syslog-ng log server

The world's first High-Speed Reliable Logging™ technology

HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the Reliable Log Transfer Protocol™
- trusted log transfer and storage

Using the FreeBSD's Procstat API in a Web Context

DAVID CARLIER

"The procstat utility displays detailed information about the processes identified by the pid arguments, or if the -a flag is used, all processes. It can also display information extracted from a process core file, if the core file is specified as the argument."

Source: http://www.freebsd.org/cgi/man.cgi?procstat

What you will learn…

- FreeBSD's procstat API
- C++ web development frameworks

What you should know…

- Programming basics
- PHP Language

Among the numerous specific features of FreeBSD, there is a famous command line to dump the statistics of the various current processes, procstat. Its internal API is fortunately exposed via the well named libprocstat library. Let's imagine we want to display it via a web page so for this article, we're going to use CppCms, one of the good quality C++ web development frameworks with the current FreeBSD 10.2 release version.

Procstat API

The list of the available functions can be viewed in this page *https://www.freebsd.org/cgi/man.cgi?query=libprocstat&sektion=3&apropos=0&manpath=FreeBSD%2010.0-RELEASE*.

We just need to include the necessary headers and link our application to the shared library libprocstat, simply. For our basic procstat service, we will expose the pids, the paths of the processes and the owners of those.

CppCms

We could have used a usual full PHP solution, calling procstat utility via a system call, possibly parsing the output and displaying it. However, doing web development via low level languages is also possible especially in the embedded environments where the resources usage count.

CppCms has a package, so pkg install cppcms (or via the ports) is sufficient. This framework has a lot of useful features; session handling, caching, native encoding handling. For our basic usage, we'll use their advanced template system with the addition of jQuery to make it more appealing.

Content

Let's start with the template's content. For this purpose we need a C++ prototype and a CppCms template file.

proclist.h :

```
#include <cppcms/view.h>
```

```
#include <vector>
// Just a plain struct to hold a specific process data
struct Procinfo {
        pid_t pid;
        std::string pathName;
        std::string args;
        std::string userName;
        std::string userFullName;
        std::string userHome;
};
// This class will be used by the template's file
// The main CppCms app will fill in the list of processes
   before the template's rendering
namespace content {
        struct ProcinfoContent : public cppcms::base_content {
                std::vector<Procinfo> pinfos;
        };
}
```

ProcinfoContentSkin.tmpl

```
// For who has experienced various templates solution for
   Java, PHP and so on, some parts seem pretty familiar
<% c++ #include „proclist.h" %> => We include simply our
   C++ prototype here
<% skin ProcinfoContentSkin %> => Useful when the template
   are shared libraries
<% view ProcinfoContent uses content::ProcinfoContent %>
<% template render() %>
<html>
        <head>
        <link rel="stylesheet" href="//jqueryui.com/
   jquery-wp-content/themes/jquery/css/base.css?v=1">
        <link rel="stylesheet" href="//jqueryui.com/
   jquery-wp-content/themes/jqueryui.com/style.css">
        <script src="//code.jquery.com/jquery-
   1.10.2.js"></script>
        <script src="//code.jquery.com/ui/1.11.4/jquery-
   ui.js"></script>
        <script type="text/javascript">
        $(function() {
                        $(„tbody").sortable();
                        $(„tbody").disableSelection();
        });
        </script>
        </head>
        <body class="jquery-ui page-template-default">
        <h1>Processes statistics</h1>
        <div class="container">
        <div id="content-wrapper">
        <div id="content">
        <table class="ui-sortable">
        <tr>
                <th>PID</th>
                <th>PATH</th>
                <th>ARGUMENTS</th>
                <th>OWNER</th>
        </tr>
        <tbody>
        <% foreach info in pinfos %> => Iterate through
   the pinfos member of the content's class ...
        <% item %>
        <tr> => ... then ,echoing' each field of a Procinfo struct
                <td class="ui-state-default ui-sortable-
   handle"><%= info.pid %></td>
                <td class="ui-state-default ui-sortable-
   handle"><%= info.pathName %></td>
                <td class="ui-state-default ui-sortable-
   handle"><%= info.args %></td>
                <td class="ui-state-default ui-sortable-
   handle"><%= info.userName %> (<%= info.userFullName %>)
   <%= info.userHome %></td>
        </tr>
        <% end %>
        <% end %>
        </tbody>
        </table>
        </div>
        </div>
        </div>
        </body>
</html>
<% end template %>
<% end view %>
<% end skin %>
```

Application

cppcms_procstat.cc :

```
// And finally the most important, the CppCms's application ...
#include <cppcms/application.h>
#include <cppcms/applications_pool.h>
#include <cppcms/service.h>
#include <cppcms/http_response.h>

#include <iostream>
#include <sstream>
#include <stdlib.h>

#include <kvm.h>
#include <sys/param.h>
#include <sys/queue.h>
#include <sys/socket.h>
#include <sys/sysctl.h>
#include <sys/types.h>
#include <sys/user.h>
```

```
#include <pwd.h>
#include <libprocstat.h>

#include „proclist.h”
class Procstat : public cppcms::application {
private:
        procstat *ps;
        content::ProcinfoContent pc;
public:
        Procstat(cppcms::service &srv) :
    cppcms::application(srv) {
       // We’re opening the processes info via the inter-
    nal sysctl system
      // There are other ways, via a kernel’s core dump file
    or via kvm ...
                ps = procstat_open_sysctl();
        }
        ~Procstat() {
                procstat_close(ps);
        }
        virtual void main(std::string url);
};

int
kp_compare(const void *a, const void *b) {
        const kinfo_proc *ka = reinterpret_cast<const
    kinfo_proc *>(a);
        const kinfo_proc *kb = reinterpret_cast<const
    kinfo_proc *>(b);

        if (ka->ki_pid < kb->ki_pid)
                return -1;
        else
                return 1;
}

void
Procstat::main(std::string) {
        unsigned int ct;
        int i;
    // we just get the processes information w/o their
    thread IDS though ...
    // We could get also only a specific group of processes
    per TTY or user etc ...
        kinfo_proc *kp = procstat_getprocs(ps, KERN_PROC_
    PROC, 0, &ct);
        if (kp == NULL)
                return;
        pc.pinfos = std::vector<Procinfo>();
        qsort(kp, ct, sizeof(*kp), kp_compare); // As the
    processes list is not ordered, we do per PID
        for (i = 0; i < ct; i ++) {
                char path[PATH_MAX];
                procstat_getpathname(ps, &kp[i], path,
    sizeof(path));
                if (strlen(path) > 0) {
                        Procinfo pi;
                        pi.pid = kp[i].ki_pid;
                        pi.pathName = std::string(path);

                        std::stringstream ss;
        // Here we get the possible arguments the process
    were called with ...
        // args NULL terminated list pointer will be freed
    by procstat_close later
                        char **args = procstat_getargv(ps,
    &kp[i], 0);
                        char **pargs = args;
        // pargs[0] == path here, so it is bypassed (hence
    we could have just used       procstat_getargv ...)
                        while (*++pargs)
                                ss << " " << *pargs;

                        pi.args = std::string(ss.str());
                        passwd pw, *res;
                        memset(&pw, 0, sizeof(pw));
                        char buf[1024];
         // Just to get more “human readable” process’ user info
                        if (getpwuid_r(kp[i].ki_ruid, &pw,
    buf, sizeof(buf), &res) == 0) {
                                pi.userName =
    std::string(pw.pw_name);
                                pi.userFullName =
    std::string(pw.pw_gecos);
                                pi.userHome =
    std::string(pw.pw_dir);
                        }

                        pc.pinfos.push_back(pi);
                }
        }
        procstat_freeprocs(ps, kp); // Important to free
    the processes information
        render(„ProcinfoContent”, pc); // Finally render-
    ing the related template ...
}
int
main(int argc, char *argv[]) {
        try {
                cppcms::service srv(argc, argv);
                srv.applications_pool().mount(
                     cppcms::applications_factory<Procstat>()
                );
```

```
        // Now our server is listening to client's requests ...
                srv.run();
        } catch (std::exception const &ex) {
                std::cerr << ex.what() << std::endl;
        }
        return 0;
}
```

Configuration

CppCms uses the popular JSON format for the configuration file as follows for our example ...

config.json :

```
{
        "service": {
                "api": "http",
                "ip": "ip address to listen",
                "port": 8180
        },
        "http": {
                "script_names": [ "/procstat" ]
        }
}
```

The possibilities of configuration are pretty rich, here we're using the internal web server, but in production it might be preferable to configure in FastCGI mode and allowing a genuine web server, like Nginx, handling the client's connections ...

```
{
        "service": {
                "api": "fastcgi",
                "socket": "" <path of the unix socket>,
        },
        "http": {
                "script_names": [ "/procstat" ]
        }
}
```

If we planned to compile the template as a shared library, we would need also to declare it in our config. For more precise information, please read this page: *http://cppcms.com/wikipp/en/page/cppcms_1x_config*.

Compilation

First, we need to "compile" the template file into a C++ code via a CppCms utilily.

```
cppcms_tmpl_cc ProcinfoContentSkin.tmpl -o
   ProcinfoContentSkin.cc
```

Then compiling our CppCms' application with this template. Indeed, for the sake of simplicity and as we have only one template, we compile it statically.

```
c++ -g -O2 -I/usr/local/include -L/usr/local/lib -o
   cppcms_procstat cppcms_procstat.cc ProcinfoContentSkin.
   cc -lcppcms -lbooster -lprocstat
```

I would advise to use at least a Makefile. The booster's library is necessary for the template's system otherwise it is also possible to render HTML content directly at the application level via an usual C++ stream like here:

```
void
Procstat::main(std::string) {
...
response().out()<<
        "<html>\n<body>\n"
        " <h1>Processes statistics</h1>\n";
...
}
```

Test

Once compiled, we can finally launch our CppCms's application.

```
./cppcms_procstat -c config.json
```

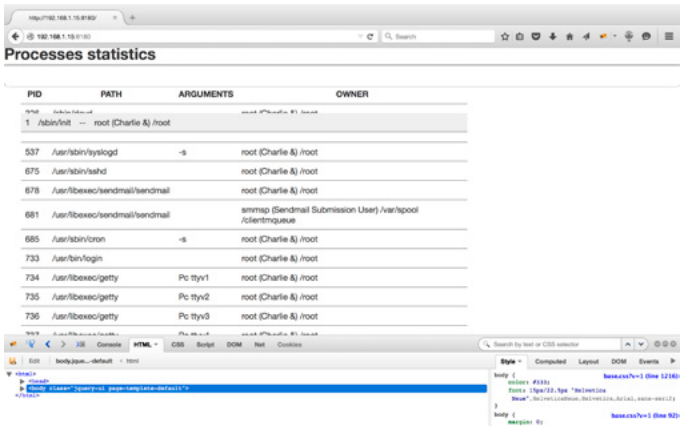


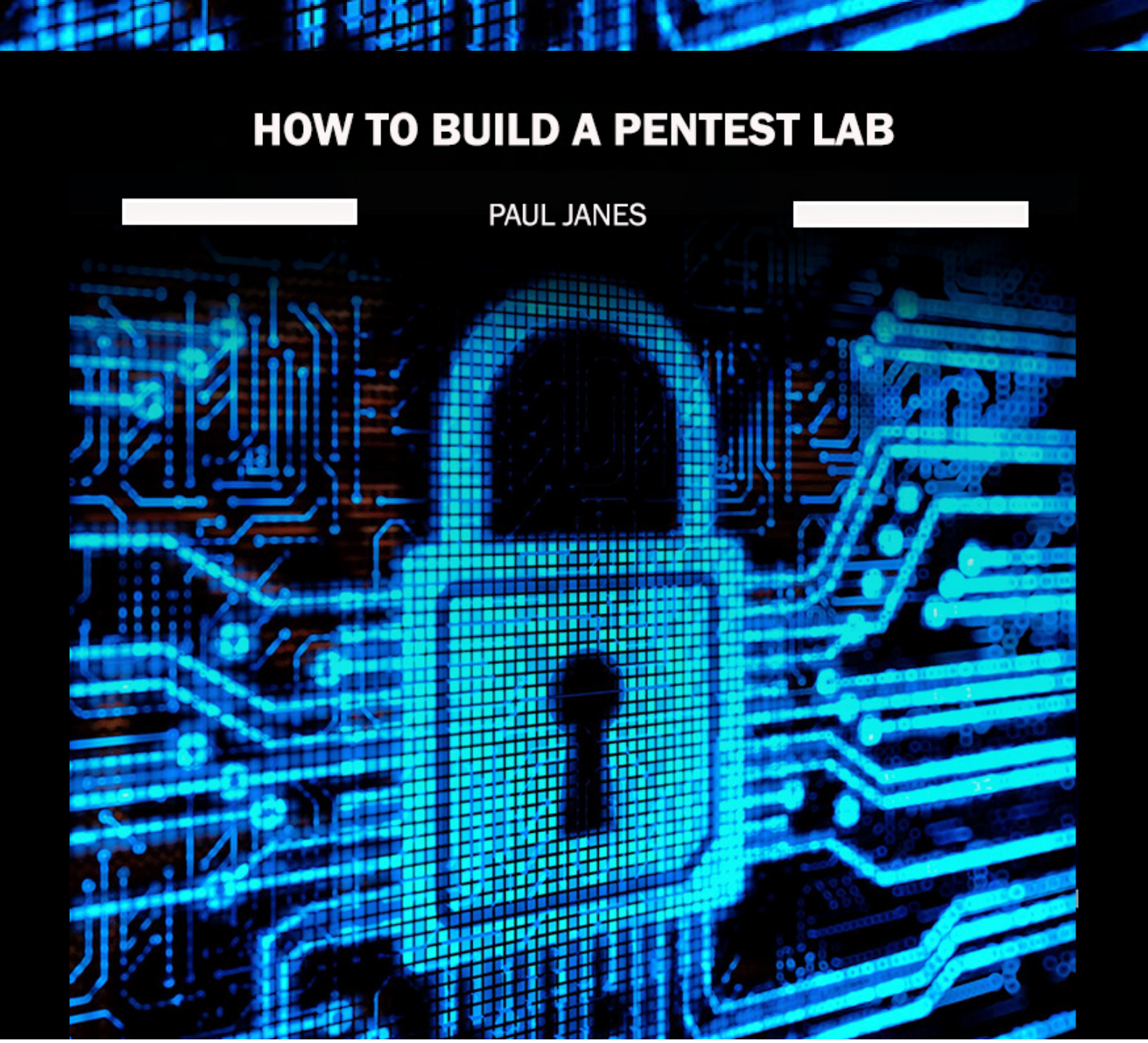
Figure 1. *Our sortable list of processes*

Conclusions

This is it, we can now read the processes list and rearrange the order in a fancy manner. There is a lot of room for improvements, hopefully, that might give some ideas to you, Readers. I hope at least, that will give you also the curiosity to dig more into the procstat's API.

ABOUT THE AUTHOR

David Carlier has been working as a software developer since 2001. He used FreeBSD for more than 10 years and starting from this year, he became involved with the HardenedBSD project and performed serious developments on FreeBSD. He worked for a mobile product company that provides C++ APIs for two years in Ireland. From this, he became completely inspired to develop on FreeBSD.



Enroll to BUILD YOUR OWN PENTEST LAB online course and learn how to create your own pentest lab.

This course covers various virtualization software and penetration testing tools like Kali Linux, Nessus, Metasploit, Metasploitable, Nmap, and others.

Through practical hands-on labs, you will be able to not only identify systems but also identify their vulnerabilities.

All in pure practice.

In case of any questions please contact:

joanna.kretowicz@eforensicsmag.com

Course Plan:

Pre-Course Material

- Why Do I Need a Pen Test Lab
- Definitions
- Creating Directory Structure For the Course
- Download Virtual Images
- Acquire Nessus Licenses

Module 1 The Build

- Definitions
- Some Basic Linux Commands You Need to Know

Software

- Installation of VMPlayer and Virtual Box. You Decide, We Will Cover Both.
- Setup of Our Penetration Testing System – Kali Linux Distribution
- Setup a Linux Client as a Virtual Machine
- Setup Our First Vulnerable Machine Metasploitable2
- Setup Our Second Vulnerable Machine Bee-box (BWAMP)

Exercises

- Overview of Virtual Machine Settings
- Run the Basic Linux commands
- Upgrade Kali Linux Distribution

Module 2 Port Scanning

- Nmap and Zenmap Installation
- Nmap Basic Scanning
- ZenMap Basic Scanning
- Metasploitable Dnmap Scanning

Exercises

- Run Nmap Scans against Ubuntu
- Run Zenmap Scans Against Metasploitable2
- Run Dnmap Scans Against Host

Module 3 Vulnerability Scans

- Installation and Licensing of Nessus Vulnerability Scanner
- Installation of Netsparker Web Vulnerability Scanner
- Basic Nessus Scanning
- Basic Netsparker Scanning
- Intermediate Nmap Scans

Exercises

- Run a Nessus Scan Against Metasploitable2
- Run a Netsparker Scans Against Bee-Box (BWAMP)
- Run a Nessus Scan Against Ubuntu

Module 4 Advanced Scanning and Reporting

- Nessus Advanced Scans
- Netsparker Advanced Scans
- Nmap Advanced Scans
- Metasploit Reporting
- Review Other Resources Available to You…
- Where Do I Get Virtual Machines

Exercises

- Create a Metasploit Report Combining Nessus and Dnmap Scans
- Run an Advanced Nessus Scan Against Metasploitable 2
- Run an Advanced Netsparker Scan Against Bee-Box (BWAMP)

If you have any questions or just want to get to know us better feel free to contact me at joanna.k@eforensicsmag.com or just answer this email

Get 10% discount on our magazines and online courses. Insert the code and use it at check-out

10eForSe07

Code is valid till the end of July

FreeNAS vs TrueNAS

BRETT DAVIS

"What's the difference between TrueNAS and FreeNAS? Is TrueNAS just FreeNAS installed on a server?" If you look at the software feature list, there aren't a ton of differences. So really….what's the difference?

The first difference is the software delivery method: TrueNAS is a purpose-built storage appliance while FreeNAS is freely-downloadable software that requires the user to understand storage well enough to select the correct hardware that is appropriate for their application.

1. TrueNAS is commercially-supported, while FreeNAS is community-supported.
2. There are performance and usability optimizations in TrueNAS that are specific to the hardware we use and therefore aren't included with FreeNAS.
3. High-Availability (failover) is hardware-dependent and only available in TrueNAS.

But, perhaps more critical to understand than the "what" is the "why":

We make FreeNAS for when storage is non-critical

There are certainly many storage applications that don't require professional support. Applications like home storage, simple office file servers, tertiary backups, home streaming media servers, scratch space, storage experimentation, or any other application where data is fungible; FreeNAS can be the perfect solution for all of them.

We make TrueNAS for when storage is critical

Storage downtime can equal an instant loss of revenue, making reliable storage a painstaking process – a process that requires careful consideration, deep hardware and storage knowledge, and countless hours of testing – certainly eons more difficult than the Software Defined Storage crowd would want you to believe. It took us nearly two years to select, design, test, and qualify the myriad hard-

ware components that go into TrueNAS, which is a purpose-built appliance – meaning software coupled with custom hardware – designed for its one specific application: critical storage. Compared to a user-built system that your software vendor knows nothing about, the appliance platform is inherently easier to support when things don't go your way, because your software vendor is your hardware vendor as well. And, when storage is this important to your business, it's imperative to have a Support Team at arm's length who can resolve any issue that may arise without having to first wrap their heads around the hardware platform you've built.

We make FreeNAS for Open Source flexibility

For those that have the expertise and the spare time to build and support their own solutions, or for those that want to tinker and learn about storage, FreeNAS is freely-available and unencumbered by license restrictions. The FreeNAS Project has a mature community and a team of developers dedicated to providing the best (open-source) software defined network file storage solution in the world. All we ask in return is that you enjoy the software and contribute when and where you can, which can be as simple as providing feedback, filing bugs, and making feature requests, or as involved as helping us write code.

We make TrueNAS for enterprise stability

Where FreeNAS is the bleeding edge, TrueNAS is the stable handle. FreeNAS is where technologies are tested and refined; therefore the software undergoes an often rapid and frequent release cycle. TrueNAS, by contrast, contains only the most stable and vetted code, keeping software updates to a minimum and the release cycle methodical.

We make FreeNAS for people who want to "DIY"

Some folks like to do it themselves. Some folks only get satisfaction when building things on their own. Some folks don't mind downtime when there's an issue and enjoy perusing the FreeNAS forums for help. Some folks have limited budgets yet still want powerful storage software. And, some folks are storage experts themselves. You're welcome, guys :)

We make TrueNAS because businesses don't want to "DIY"

Instead of buying a fleet of delivery trucks, I suppose we could purchase all the components separately, build the trucks ourselves, and fix them when things break. But, we're not a car dealership, we're a storage company. We'd probably save money up front on the cost of the bare parts but would certainly come out way behind with the time spent figuring out how to put them all together and build a functioning car, let alone the costs to maintain it! Most businesses don't have the time, available hardware, or internal support expertise for a do-it- yourself storage solution – they're busy focused on their own missions and business models. But, with a 100% software solution, you must build the server yourself. If there is a problem with the server hardware, you can't look to the software vendor for support, and vice-versa if you have hardware problems. With TrueNAS, you get one throat to choke….ours :)

We make FreeNAS because many are turning to virtualization

FreeNAS is known to work well with all major virtualization platforms, but due to the nature of the decoupled hardware, we aren't able to officially certify the software with the virtualization vendors. Therefore, if something goes haywire, the user cannot turn to the virtualization vendor for assistance and instead must rely on the FreeNAS community.

We make TrueNAS because many are turning to virtualization…and need Support

With a software-only solution you must verify that every component is on the virtualization vendors' compatibility list and when your configuration changes (such as upgrading to a new network card) you need to validate the configuration again. Most businesses can't afford the risk, so TrueNAS is officially certified to support Citrix XenServer, VMware ESXi, and Microsoft Hyper-V.

FreeNAS and TrueNAS both have their rightful places

FreeNAS is the world's most popular software defined storage OS, with more downloads and installs than any other storage software on the planet. The sheer magnitude of interest speaks volumes about its myriad applications. And, as its enterprise counterpart, TrueNAS has the performance, high-availability, functionality, and professional software support that mission-critical storage applications require.

ABOUT THE AUTHOR

Brett Davis
iXsystems Executive Vice President

UNIX Basics

SAMANVAY GUPTA

UNIX United is architecture for a distributed system based on UNIX. Any program written for a normal UNIX system can be transparently extended to exploit the richer environment of UNIX United. As it relies on having a UNIX system beneath it, the implementation of UNIX United, called the Newcastle Connection. This paper explains the basic semantics of UNIX United and is followed by that of the architecture implied by the protocol between components in a UNIX United system, network basics and of a software structure appropriate to the architecture and the protocol.

UNIX United and the Newcastle Connection were first described in [1], which contained a quite extensive survey of work on UNIX-based distributed systems and comparisons of the different approaches that have been adopted. No attempt is made to repeat such a survey in the present paper. Since that time, the two notions of UNIX United as an architecture and the Newcastle Connection as an implementation have become more distinct in our own minds, and both have evolved considerably in response to our continuing design and implementation efforts.

The purpose of this paper is twofold: to describe the semantics and architecture of UNIX United in some detail and to discuss the current state of our design and implementation. A UNIX United system is composed of a number of component UNIX systems connected by one or more communications media. In architectural terms, UNIX United is a loosely coupled collection of components for a number of reasons: it should be feasible to use both fast and slow communications media, administrators of a component should retain their autonomy in the distributed system, and any given UNIX United system should be capable of encompassing an arbitrary number of components. While UNIX United is intentionally loosely coupled in the senses described above, it paradoxically presents an extremely integrated view to its users; that of a single, albeit very large, UNIX system in which all of the normal UNIX system calls and programs exhibit exactly the same behavior when executed in the UNIX United environment as when executed in the environment of a single, isolated component. The result is that UNIX United is recursively structured [2]: the functionality of the distributed system as a whole is identical to that of its components. This not only has some interesting consequences in terms of the design of distributed computing systems, but it also implies that all existing software investments in UNIX can be retained in UNIX United, without necessarily requiring any modification to their source code or that of the UNIX kernels on the component machines. (As distributed commercially, the Newcastle Connection consists essentially of a replacement for the C language system call library, and thus programs only need to be relinked to be used in the UNIX United environment. However, we and others have also created UNIX United systems by installing the Newcastle Connection software below the physical machine kernel boundary, just “on top of” the essentially unmodified kernel. In this case, no change whatever is

required to existing programs. Clearly, this also implies that the user's perception of UNIX United is identical to his perception of UNIX itself; the advantages of this cannot be overstated. In Section II, we discuss the motivation and basic semantics of UNIX United in more detail. Section III discusses the architecture of UNIX United, or precisely how the semantics of UNIX are extended in UNIX United. Section IV describes the software structures associated with the architecture, both in terms of our implementation (the Newcastle Connection), and in terms of the remote system call protocol which is used between various processes on UNIX machines in a UNIX United system.

History Of Unix

The Unix operating system found its beginnings in MULTICS, which stands for Multiplexed Operating and Computing System. The MULTICS project began in the mid-1960s as a joint effort by General Electric, Massachusetts Institute for Technology and Bell Laboratories. In 1969, Bell Laboratories pulled out of the project. One of Bell Laboratories people involved in the project was Ken Thompson. He liked the potential MULTICS had, but felt it was too complex and that the same thing could be done in simpler way. In 1969, he wrote the first version of Unix, called UNICS. UNICS stood for Uniplexed Operating and Computing System. Although the operating system has changed, the name stuck and was eventually shortened to Unix.

Ken Thompson teamed up with Dennis Ritchie, who wrote the first C compiler. In 1973, they rewrote the Unix kernel in C. The following year, a version of Unix known as the Fifth Edition was first licensed to universities. The Seventh Edition, released in 1978, served as a dividing point for two divergent lines of Unix development. These two branches are known as SVR4 (System V) and BSD.

Ken Thompson spent a year's sabbatical with the University of California at Berkeley. While there he and two graduate students, Bill Joy and Chuck Haley, wrote the first Berkeley version of Unix, which was distributed to students. This resulted in the source code being worked on and developed by many different people. The Berkeley version of UNIX is known as BSD, Berkeley Software Distribution. From BSD came the vi editor, C shell, virtual memory, Sendmail, and support for TCP/IP.

For several years SVR4 was more conservative, commercial, and well supported. Today, SVR4 and BSD look very much alike. Probably the biggest cosmetic difference between them is the way the `ps` command functions.

What Is Unix?

UNIX is a powerful computer operating system originally developed at AT&T Bell Laboratories. It is very popular among the scientific, engineering, and academic communities due to its multi-user and multi-tasking environment, flexibility and portability, electronic mail and networking capabilities, and the numerous programming, text processing and scientific utilities available. It has also gained widespread acceptance in government and business. Over the years, two major forms (with several vendor's variants of each) of UNIX have evolved: AT&T UNIX System V and the University of California at Berkeley's Berkeley Software Distribution (BSD). This document will be based on the SunOS 4.1.3_U1, Sun's combination of BSD UNIX (BSD versions 4.2 and 4.3) and System V because it is the primary version of UNIX available at Rice. Also available are Solaris, a System V based version, and IRIX, used by Silicon Graphics machines.

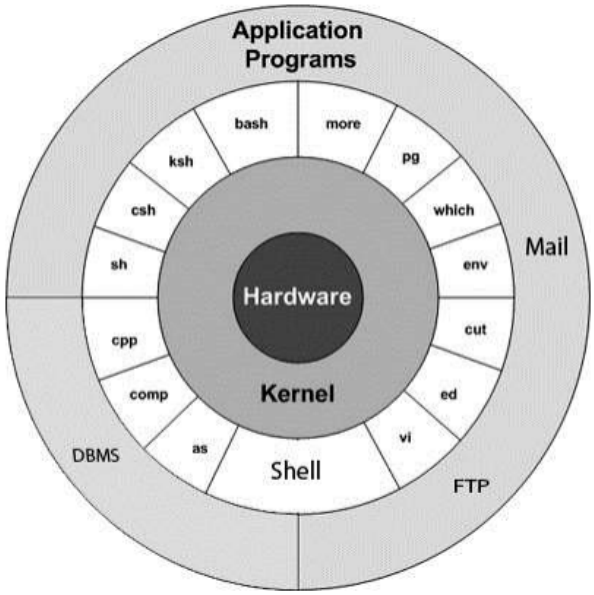


Figure 1. *Structure*

Unix Basics – Structure

The main concepts that unite all versions of UNIX are the following four basics:

- Kernel: The kernel is the heart of the operating system. It interacts with hardware and most of the tasks like memory management, task scheduling and file management.
- Shell: The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are most famous shells which are available with most of the UNIX variants.

- Commands and Utilities: There are various command and utilities which you would use in your day to day activities. cp, mv, cat and grep, etc. are a few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various optional options.
- Files and Directories: All data in UNIX is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system.

Directory Structure

The UNIX system is set up as a tree hierarchy. At the top of the tree is the root. The root is represented by the slash character. Off of the root are branches of the tree. The branches are directories.

Files or directories can be off the tree.

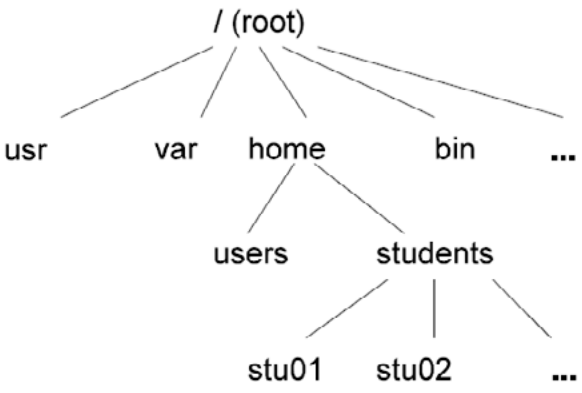


Figure 2. *Tree hierarchy*

Design: An Extensible Kernel

Early in its development, UNIX supported the notion of objects represented as file descriptors with a small set of basic operations on those objects (e.g., read, write and seek) [3]. With pipes serving as a program composition tool, UNIX offered the advantages of simple implementation and extensibility to a variety of problems. Under the weight of changing needs and technology, UNIX has been modified to provide a staggering number of different mechanisms for managing objects and resources. In addition to pipes, UNIX versions now support facilities such as System V streams, 4.2 BSD sockets, pty's, various forms of semaphores, shared memory and a mind-boggling array of IOCtl operations on special files and devices. The result has been scores of additional system calls and options […]

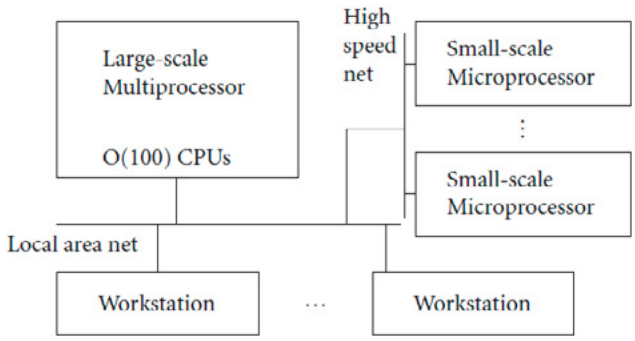


Figure 3. *Network scheme*

[…] with less than uniform access to different resources within a single UNIX system and within a network of UNIX machines. As the complexity of distributed environments and multiprocessor architectures increases, it becomes increasingly important to return to the original UNIX model of consistent interfaces to system facilities. Moreover, there is a clear need to allow the underlying system to be transparently extended to allow user-state processes to provide services which, in the past, could only be fully integrated into UNIX by adding code to the operating system kernel. Mach takes an essentially object-oriented approach to extensibility. It provides a small set of primitive functions designed to allow more complex services and resources to be represented as references to objects. The indirection thus provided allows objects to be arbitrarily placed in the network (either within a multiprocessor or a workstation) without regard to programming details. The Mach kernel abstractions, in effect, provide a base upon which complete system environments may be built. By providing these basic functions in the kernel, it is possible to run varying system configurations on different classes of machines while providing a consistent interface to all resources. The actual system running on any particular machine is a function of its servers rather than its kernel.

The Mach kernel supports four basic abstractions:

- A task is an execution environment in which threads may run. It is the basic unit of resource allocation. A task includes a paged virtual address space and protected access to system resources (such as processors, port capabilities and virtual memory). The UNIX notion of a process is, in Mach, represented by a task with a single thread of control.
- A thread is the basic unit of CPU utilization. It is roughly equivalent to an independent program counter operating within a task. All threads within a task share access to all task resources.

- A port is a communication channel – logically a queue for messages protected by the kernel. Ports are the reference objects of the Mach design. They are used in much the same way that object references could be used in an object oriented system. Send and Receive are the fundamental primitive operations on ports.
- A message is a typed collection of data objects used in communication between threads. Messages may be of any size and may contain pointers and typed capabilities for ports.

Operations on objects other than messages are performed by sending messages to ports which are used to represent them. The act of creating a task or thread, for example, returns access rights to the port which represents the new object and which can be used to manipulate it. The Mach kernel acts in that case as a server which implements task and thread objects. It receives incoming messages on task and threads ports and performs the requested operation on the appropriate object. This allows a thread to suspend another thread by sending a suspend message to that thread's thread port even if the requesting thread is on another node in a network.

The design of Mach draws heavily on CMU's previous experience with the Accent [4] network operating system, extending that system's facilities into the multiprocessor domain:

- The underlying port mechanism for communication provides support for object-style access to resources and capability based protection as well as network transparency,
- All systems abstractions allow extensibility both to multiprocessors and to networks of uniprocessor or multiprocessor nodes,
- • Support for parallelism (in the form of tasks with shared memory and threads) allows for a wide range of tightly coupled and loosely coupled multiprocessors and
- Access to virtual memory is simple, integrated with message passing, and introduces no arbitrary restrictions on allocation, deallocation and virtual copy operations and yet allows both copy-on-write and read-write sharing.

The Mach abstractions were chosen not only for their simplicity but also for performance reasons. A performance evaluation study done on Accent demonstrated the substantial performance benefits gained by integrating virtual memory management and interprocess communication. Using similar virtual memory and IPC primitives, Accent was able to achieve performance comparable to UNIX systems on equivalent hardware [5]

Accessing A Unix System

There are many ways that you can access a UNIX system. If you want the fullest possible access to the computer's commands and utilities, you must initiate a login session. The main mode of initiating a login session to a UNIX machine is through a *terminal*, which usually includes a keyboard, and a video monitor. When a terminal establishes a connection to the UNIX system, the UNIX kernel runs a process called a *tty* to accept input from the terminal, and send output to the terminal. When the tty process is created, it must be told the capabilities of the terminal, so it can correctly read from, and write to, the terminal. If the tty process receives incorrect information about the terminal type, unexpected results can occur.

The Unix Processes

A process is the flow of execution of a set of program instructions and owns, as a system entity, the necessary resources. Some operating systems, such as z/OS, call the basic unit of execution a job or task. In UNIX, it is called a process. In the UNIX kernel, anything that is done, other than autonomous operations, is done by a process that issues system calls. Processes often spawn other child processes, using, for instance, the `fork()` system call, which usually run in parallel with their parent process. These are usually subtasks which, when they are finished, terminate themselves. All UNIX processes have an owner. Typically, the human owner of a process is the owner of the account whose login process spawned the initial process parent of the process chain currently executing. The child process inherits the file access and execution privileges belonging to the parent.

Signals

Signals are designed for processes to communicate with each other and with the kernel. The signalling capability is provided by the operating system and is used, for instance, to inform processes of unexpected external events, such as a timeout or forced termination of a process. A signal consists of a prescribed message with a default action embedded in it. There are different types of signals in UNIX, and each type is identified with a number.

Console

Every UNIX system has a main console that is connected directly to the machine. The console is a special type of terminal that is recognized when the system is started. Some Unix system operations must be performed at

the console. Typically, the console is only accessible by the system operators and administrators.

Dumb Terminals

Some terminals are referred to as "dumb" terminals because they have only the minimum amount of power required to send characters as input to the UNIX system, and receive characters as output from the UNIX system. Personal computers are often used to emulate dumb terminals, so that they can be connected to a UNIX system. Dumb terminals can be connected directly to a UNIX machine, or may be connected remotely, through a modem, a terminal server, or other network connection.

Smart Terminals

Smart terminals, like the X terminal, can interact with the UNIX system at a higher level. Smart terminals have enough on-board memory and processing power to support graphical interfaces. The interaction between a smart terminal and a UNIX system can go beyond simple characters to include icons, windows, menus, and mouse actions.

Network-Based Access Modes

UNIX computers were designed early in their history to be network-aware. The fact that UNIX computers were prevalent in academic and research environments led to their broad use in the implementation of the Department of Defense's Advanced Research Projects Administration (DARPA) computer network. The DARPA network laid the foundations for the Internet.

FTP

The FTP (File Transfer Protocol) provides a simple means of transferring files to and from a UNIX computer. FTP access to a UNIX machine may be authenticated by means of a username and password pair, or may be anonymous. An FTP session provides the user with a limited set of commands with which to manipulate and transfer files.

TELNET

Telnet is a means by which one can initiate a UNIX shell login across the Internet. The normal login procedure takes place when the telnet session is initiated.

HTTP

The HTTP protocol has become important in recent years because it is the primary way in which the documents that constitute the World Wide Web are served. HTTP servers are most often publicly accessible. In some cases, access to documents provided by HTTP servers will require some form of authentication.

HTTPS

A variation of HTTP that is likely to become increasingly important in the future. The "S" stands for "secure." When communications are initiated via the HTTPS protocol, the sender and recipient use an encryption scheme for the information to be exchanged. When the sending computer transmits the message, the information is encrypted so that outside parties cannot examine it. Once the message is received by the destination machine, decryption restores the original information.

SHELLS

Processes operate in the context of a *shell*.

The shell is a command interpreter which:

- Interprets built in characters, variables and commands
- Passes the results on to the kernel. The *kernel* is the lowest level of software running. It controls access to all hardware in the computer.

sh: Bourne Shell
_ Developed by Stephen Bourne at AT&T Bell Labs
csh: C Shell
_ Developed by Bill Joy at University of California, Berkeley
ksh: Korn Shell
_ Developed by David Korn at AT&T Bell Labs
_ backward-compatible with the Bourne shell and includes many features of the C shell
bash: Bourne Again Shell
_ Developed by Brian Fox for the GNU Project as a free software replacement for the Bourne shell (sh)
_ Default Shell on Linux and Mac OSX
_ The name is also descriptive of what it did, bashing together the features of sh, csh and ksh tcsh: TENEX C Shell
_ Developed by Ken Greer at Carnegie Mellon University
_ It is essentially the C shell with programmable command line completion, command-line editing, and a few other features

There are many shells! Common features that all shells have:

- Command execution.
- Redirection of input and output.
- Piping.
- Wildcard expansion.
- Process control.
- Command recall and editing.
- *Turing-complete* (except for the memory part).

Shell scripts

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by a pound sign, #, describing the steps. There are conditional tests, such as value A is greater than value B, loops allowing us to go through massive amounts of data, files to read and store data, and variables to read and store data, and the script may include *functions*. We are going to write a lot of scripts in the next several hundred pages, and we should always start with a *clear goal* in mind. By clear goal, we have a specific purpose for this script, and we have a set of expected results. We will also hit on some tips, tricks, and, of course, the gotchas in solving a challenge one way as opposed to another to get the same result. All techniques are not created equal. Shell scripts and functions are both *interpreted*. This means they are not compiled. Both shell scripts and functions are ASCII text that is read by the Korn shell command interpreter. When we execute a shell script, or function, a command interpreter goes through the ASCII text line by line, loop by loop, and test by test and executes each statement, as each line is reached from the top to the bottom.

Shells contain:

- Variables
- Loops
- Conditional statements
- Input and Output
- Built in commands
- Ability to write functions

Specifying the shell to be used:

On the first line of the file:

- Implicitly
  - blank line – Bourne shell
  - # in column 1 – C shell
- Explicitly
  - #!/bin/sh – Bourne shell
  - #!/bin/csh – C shell

Directory Commands

After logging into the system, the current directory is your home directory. So for the account stu01 the current directory would be `/home/students/stu0`1. To view what the current directory is, use the pwd command:

```
$ pwd
```

To create a new directory off of the home directory uses the command mkdir.

```
$ mkdir newdir
```

To view a listing of the contents of the current directory use the command ls.

```
$ ls
```

For a directory listing that gives more information use the command:

```
$ ls -l
```

To view hidden files those don't normally show up with an ls use the command:

```
$ ls -la
```

To change the current directory to the new directory that was just created use the change directory command `cd`.

```
$ cd newdir
```

The newdir directory is down one level in the tree from the home directory for stu01. Check to see what directory is current:

```
$ pwd
```

In this directory, files could be stored or additional sub directories could be created.

To move back up one directory use the command:

```
$ cd ..
```

The dot dot represents the current directory.

To rename a directory use the move command mv.

```
$ mv newdir newname
```

The Unix File System

The UNIX file system hosts the collection of files accessed by the processes running in the system and is in charge of the logical representation of the data to the requesting entities. The file system has therefore both a logical and physical dimension.

The logical file system

The logical file system is in charge of the hierarchy of connected directories and files as they are shown to the users. The UNIX file system is logically arranged as a tree, actually inverted with the root, named "/", at the top. All files are logically contained within the root directory. See the

example shown in Figure 4, where the shaded boxes represent directories, while the unshaded boxes represent files. A file or directory is located in the file system tree using a "path name"; `/etc/profile` or `/u/dirA/dirA1/Dominique` are path names. Note that UNIX is a case-sensitive operating system; therefore a file called "ABC" is different from a file called "abc".

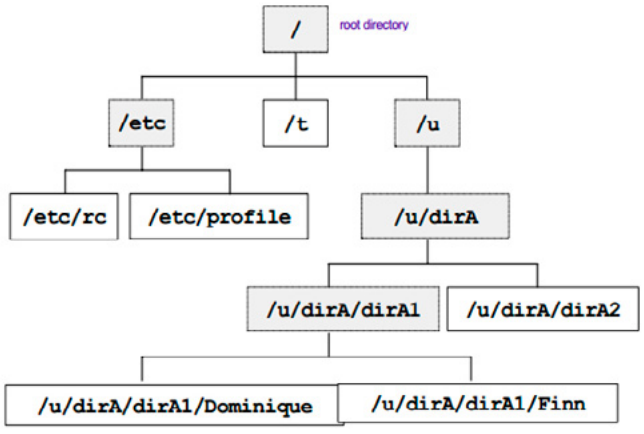


Figure 4. *Logical File System*

The physical file system

The physical file system, as the name implies, is in charge of the physical arrangement of data and control information about the physical media. The physical file system operates with control blocks such as the superblock, inodes, and data blocks. The superblock holds the control information for the system. Inodes contain similar information for individual files. The data blocks hold the data that makes up the information in the files.

Conclusion

UNIX provides bothappropriate semantics for a general-purpose distributed system and appropriate mechanisms and interfaces for this system to be constructed merely by adding a comparatively simple transparent subsystem to UNIX. The design philosophy we employed was, at the outset, little more than an active concern for structure and generality, and, more particularly, a liking for recursive constructs (dating back to work at Newcastle on recursive virtual machines [6], if earlier). However, as a result of our work on the Connection, these ideas on recursive system structuring have become much more well defined, in our own minds at least, and have enabled us to separate carefully issues concerned with constructing a distributed system from those concerned with taking advantage of the fact that it is distributed, for example, in order to provide increased reliability, availability, and/or security. This is not to say that we have simply ignored all such issues. Rather we have investigated, and in several cases already implemented, various separate but complementary reliability and security mechanisms, each of which can simply be added to a UNIX United system, without requiring modifications to the code of either UNIX or the Connection [7], [8], and [9]. (This work is surveyed in [10], as part of a general account of our ideas on recursive structuring.)

It would be inappropriate to end these concluding remarks without an explicit acknowledgment of our debt to UNIX and its original creators–it has its deficiencies, of course, both as a centralized system, and as the basis of a general-purpose distributed system. Nevertheless, we have found its facilities, particularly at the system call level, and the style of system design that it exemplifies a veritable inspiration. Such simplicity and generality of mechanism as we have been able to achieve undoubtedly owes much to this source.

Reference

[1]. D. R. Brownbridge, L. F. Marshall, and B. Randell, "The Newcastle Connection-or UNIXes of the world unite!" Software–Practice and Experience, vol. 12, no. 12, pp. 1147- 1162, Dee. 1982.

[2]. B. Randell, "Recursively structured distributed computer systems," in Proc. Symp. on Reliability in Distributed Software and Database Systems, pp. 3-11, Oct. 1983.

[3]. D. M. Ritchie and K. Thompson. The Unix time sharing system. Communicationsof the ACM, 17(7):365–375, July 1974.

[4]. R. F. Rashid and G. Robertson. Accent: A communication oriented network operating system kernel. pages 64–75. ACM, December 1981.

[5]. R. Fitzgerald and R. F. Rashid. The integration of virtual memory management and interprocess communication in accent. ACM Transactions on Computer Systems, 4(2), May 1986.

[6]. H. C. Lauer and D. Wyeth, "A recursive virtual machine architecture," in Proc. ACM Workshop on Virtual Computer Systems, pp. 113-116, Mar. 1976. Also available at University of Newcastle upon Tyne, Computing laboratory, Tech. Rep. TR54.

[7]. J. A. Anyanwu, "A reliable stable storage system for UNIX," Software–Practice and Experience, vol. 15, no. 10, pp. 973-990, Oct. 1985.

[8]. J. A. Anyanwu and L. F. Marshall, "A crash resistant UNIX file system," SoftwarePractice and Experience, vol. 16, no. 2, pp. 107-118, Feb. 1986.

[9]. J. M. Rushby and B. Randell, "A distributed secure system," IEEE Computer, vol. 16, no. 7, pp. 55-67, July 1983. Also available at University of Newcastle upon Tyne, Computing laboratory, Tech. Rep. TR182.

[10]. B. Randell, "Recursively structured distributed computer systems," in Proc. Symp. on Reliability in Distributed Software and Database Systems, pp. 3-11, Oct. 1983.

ABOUT THE AUTHOR

Samanvay Gupta Security Researcher and Analyst in Hicube InfoSec, Cyber Security Expert, Ethical Hacker, MCITP professional, Information Security Expert, Author of 6 International Journals, Delivered workshops, seminars and trainings in different parts of the country.

Learn How To Master Big Data

BigData TECHCON

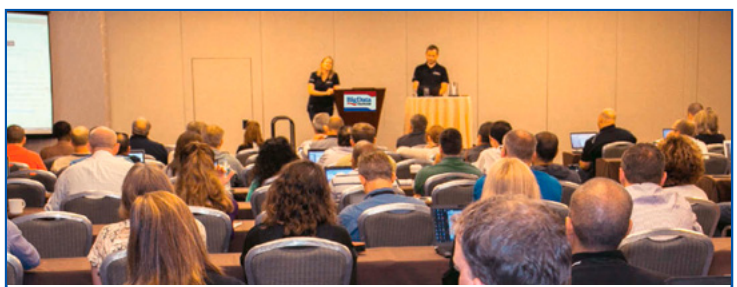
November 2-4, 2015
CHICAGO

Holiday Inn Chicago Mart Plaza River North

Choose from 55+ classes and tutorials!

Attend Big Data TechCon to get practical training on Hadoop, Spark, YARN, R, HBase, Hive, Predictive Analytics, and much more!

Take a Big Data analytics tutorial, dive deep into machine learning and NoSQL, learn how to master MongoDB and Cassandra, discover best practices for using graph databases such as Neo4j and more. You'll get the best Big Data training at Big Data TechCon!



People are talking about BigData TechCon!

Great for quickly coming up to speed in the big data landscape.
—Ben Pollitt, Database Enginee, General Electric

There was a large quantity and variety of educational talks with very few sales lectures. It was just informative and inspiring. This was the best conference ever! Get a ticket for 2015!
—Byron Dover, Big Data Engineer, Rubicon Project

www.BigDataTechCon.com

A BZ Media Event

UNIX – How To Start Terminal?

NITIN KANOIJA

UNIX is a multiuser operating system which is available in many flavours, like Oracle Solaris, HP UNIX, IBM AIX, Free BSD, and MacOS. It was developed by Ken Thompson and Dennis Ritchie at AT&T Bell Laboratories in the late 1960's. In 1978, AT&T's UNIX seventh edition was split off into Berkeley Software Distribution (BSD). This version of the UNIX environment was sent to other programmers around the country, who added tools and code to further enhance BSD UNIX.

The most important enhancement made to the OS by the programmers at Berkeley was adding networking capability. This enabled the OS to operate in a local area network (LAN). In 1988, AT&T UNIX, BSD UNIX, and other UNIX OSs were folded into what became System V release 4 (SVR4) UNIX. This was a new generation OS, which became an industry standard. The new SVR4 UNIX became the basis for not only Sun and AT&T versions of the UNIX environment, but also IBM's AIX and Hewlett-Packard's HP-UX.

UNIX was constructed with following mechanisms:

Kernel

Kernel is the core/heart of an OS and is responsible for all the processing in a computer. It manages all the physical resources of the computer, including filesystems, CPU, memory, etc.

Shell

Shell is a command interpreter and acts as an interface between the system and the user. Shell accepts the command and passes it to the kernel, which further executes the command. In Oracle Solaris 11 and Oracle Enterprise Linux, the default shell is Bourne Again Shell, which is also known as bash.

File System

A file system is a logical collection of a files and directories on a partition or a disk. It has a root directory, which further contains all files and directories in an operating system. The root directory is identified as /. Each file or directory is identified by its name and a unique identifier known as Inode number.

Process

Every program you run or execute in UNIX/Linux creates a process. When you log in to the system and start the shell, several processes will be started, depending on the associated programs in login shell. Whenever you execute a command in the shell, it will start a process, which can further start another process. In that case, the process which has started another process will be known as a parent process. You can use the following commands in UNIX/Linux to monitor and manage the process: Ps, top, prstat,

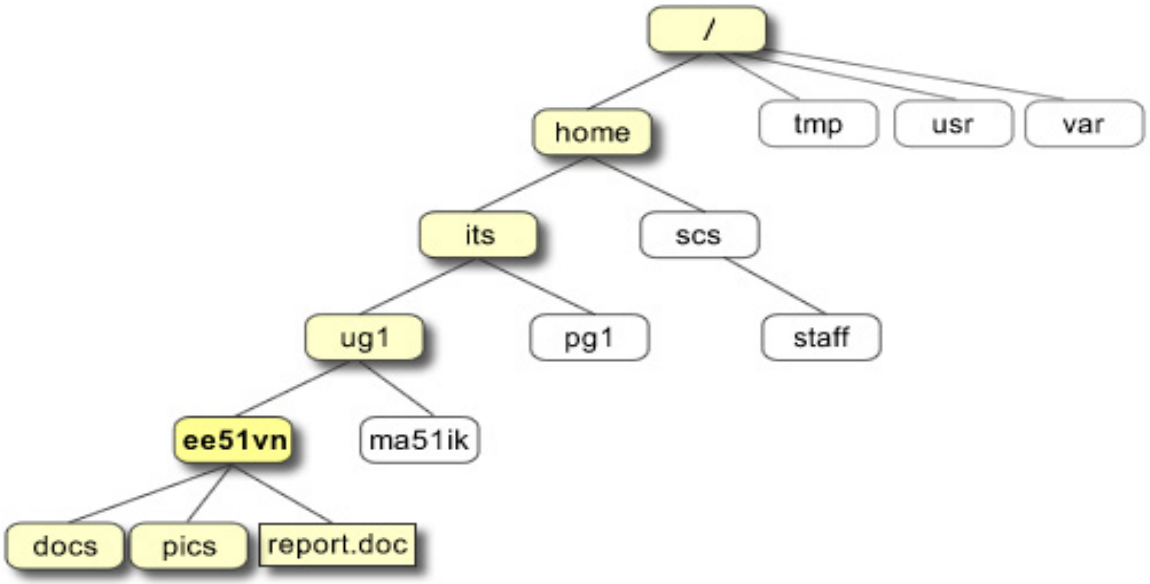


Figure 1. *Directory structure*

pgrep.

Solaris and HP UNIX are widely used flavours of UNIX. Since UNIX was developed, many features and tools have been added to different flavours of UNIX, l*ike Journaling file system, ZFS, DTrace, enhanced packaging system like IPS, Solaris Volume manager (which was earlier know as Solstice Disk Suite).*

Who should use UNIX/Linux?

Companies, or system administrators, who have big servers in their environment and need stability, scalability, security and high performance for their servers should use UNIX/Linux operating systems. UNIX/Linux operating system uses much less resources in comparison to any other operating systems. UNIX/Linux has many enhanced security features, like SELinux, IP tables, TCP wrappers, ACLs, Dtrace and many more.

How to start terminal in Oracle Solaris 11?

To open a terminal window in Oracle Solaris 11, right click on the Desktop and left click onthe ”Open Terminal” option in the menu.

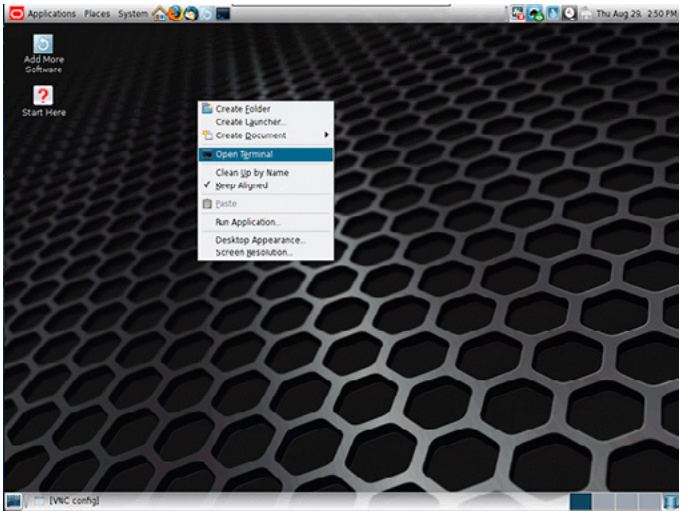


Figure 2. *Oracle Solaris 11 Desktop Menu*

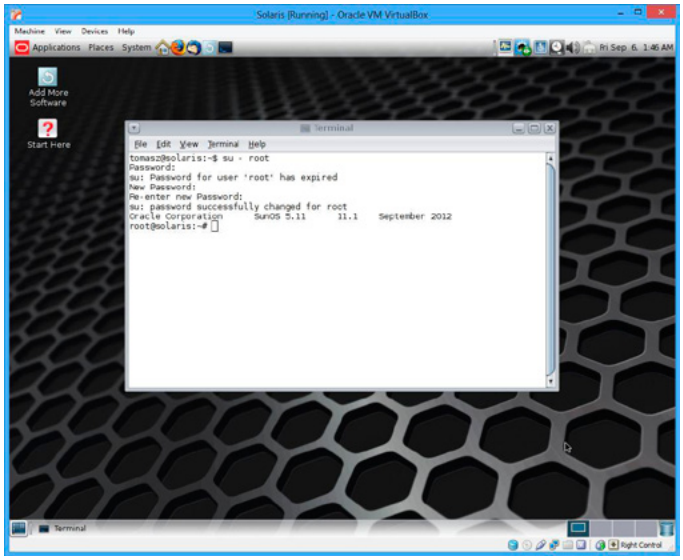


Figure 3. *Terminal window*

An Oracle Solaris 11 Terminal window will then appear with a $ prompt, and you can start entering the commands.

Oracle Solaris 11 Desktop:

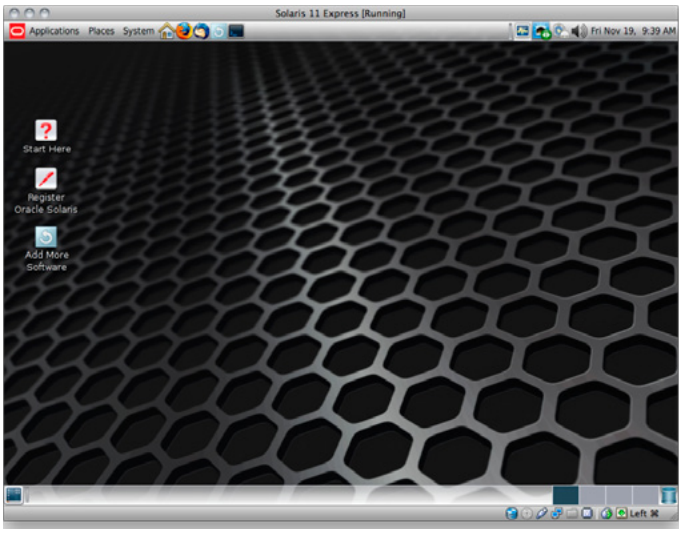


Figure 4. *Oracle Solaris 11 Desktop*

Installation Options for Oracle Solaris 11 (Flavour of UNIX)

You have several alternatives for where to install Oracle Solaris 11:

- Inside a virtual machine on top of your existing operating system
- On the bare metal (physical machine) as a standalone operating system
- On the bare metal alongside your existing operating system(s) (multiboot/dual boot scenario)

Installing Oracle Solaris 11 inside a Virtual Machine with Live CD

The easiest way to start using Oracle Solaris 11 is to install it into a virtual machine on top of the host operating system running on the physical machine. The figure below shows Oracle Solaris 11 installed on Apple OS X using Oracle VM Virtual Box.

Oracle Solaris 11 will recognize the virtualized devices that the virtual machine provides. If you run Oracle Solaris 11 in full-screen mode, you might actually forget that there's another operating system running in the background. The one drawback to this approach is that you need enough memory to run two operating systems simultaneously – a minimum of 2 GB is recommended for good performance. You should also allow a minimum of 7 GB of disk space to install the operating system in virtual machine.

Oracle VM VirtualBox is a free-to-download virtualization application that can run on Microsoft Windows, Apple OS X, Linux, and Oracle Solaris x86 as host platforms, and supports most of the flavours of Linux, like Redhat & Oracle Enterprise Linux as guest OS. It also supports Oracle Solaris as one of its many guests. Oracle makes it easy to try this approach by offering a number of pre-installed virtual machines for Oracle VM VirtualBox as appliances and VM templates that are focused towards a specific use, for example, to evaluate the developer tools that are available on Oracle Solaris 11.

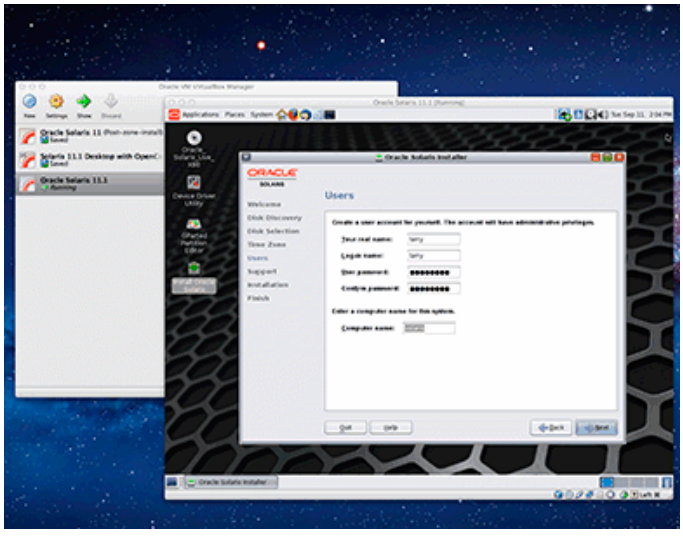


Figure 5. *Oracle Solaris on Apple OS X*

After you have booted off the Live Media, the installation process is straightforward. Simply click the *Install Oracle Solaris icon* on the desktop to launch the graphical installer, shown in Figure 6.

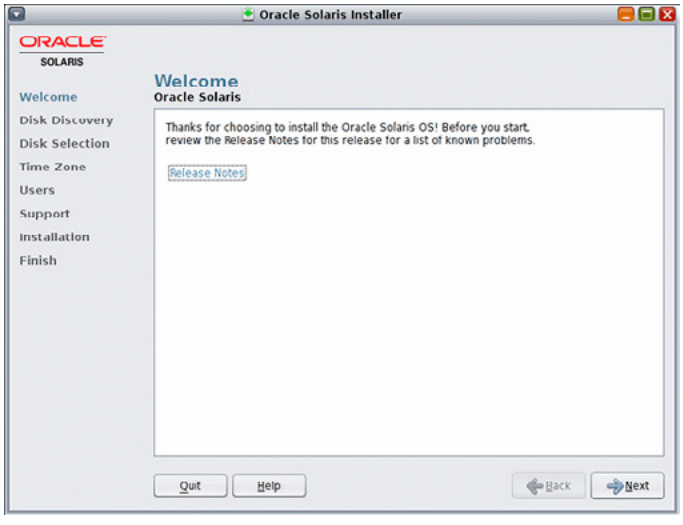


Figure 6. *The Oracle Solaris 11 Graphical Installer*

As you can see from the above Figure, the installation process is simple and asks some basic questions before installing a fixed set of packages. After Oracle Solaris has successfully been installed, you can easily customize the installation by using the Package Manager. After the installation process is complete, you can reboot into your new Oracle Solaris environment or review the Oracle Solaris installation log, as shown in Figure 7.

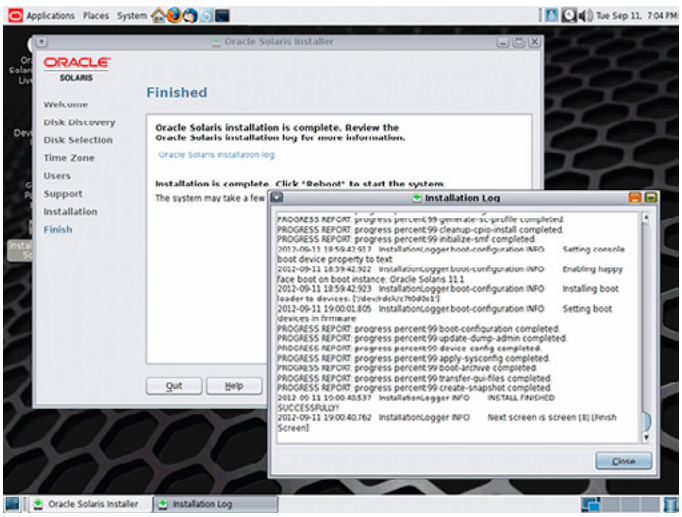


Figure 7. *Reviewing the Installation Log*

Now you are ready to launch your work.

ABOUT THE AUTHOR

Nitin Kanoija has 8+ years of experience in IT industry with core expertise in Unix/Linux and Veritas. He is currently working as Senior Corporate Trainer with Koenig Solutions Ltd. Nitin possesses vast experience on Unix/Linux, Oracle Virtualization & Clustering technologies and has also handled several projects which demand in-depth knowledge of Unix/Linux and clustering. Nitin is Sun Certified System Administration Certification (SCSA) & Sun Certified Network Administration Certification (SCNA).

How About Some Raspberry Pi?

JERRY CRAFT

In early 2006, Eben Upton was working with undergraduate admissions in computer science as a PhD Candidate for the University of Cambridge. Working in admissions, he was hoping to find kids who were used to playing around with computers, but instead discovered something different. The love for figuring out how a computer functioned wasn't part of the college application. Eben discovered kids were no longer writing programs and taking apart circuit boards. Instead, they were playing video games or using the family computers to update MySpace/Facebook posts. Kids didn't have access to a computer they could blow up or really get into and discover how a computer functions. The hacking instinct was gone. Instead, kids going into college for computer science were "..consumers of computers." (Mann)

Eben decided that, in order to change this, there needed to be a simple low cost alternative for kids to use and discover a different side of computing, the side of computers that Eben, and anyone prior to 1995, grew up discovering. Eben wanted to help kids learn about programming, circuitry, and the basics they had been missing in the applications he was reviewing. Eben decided to build a cheap single board computer called Raspberry Pi to facilitate that discovery. During his growing up, he discovered how to take apart computers, build programs, and discover how the systems work from machine language to basic electronics (Figure 1).

Figure 1. *Eben Upton*

I too had a similar experience growing up. I personally came to computers in the 80's when I was 16. My first computer was a Commodore VIC 20. It had no hard drive because at that time they were too expensive. Likewise, it had no floppy drive, tape drive, and it would only boot to ROM BASIC. My family was too poor to buy the computer so I spent a year working to save up enough money to buy this $100 system. But I did it, and when I brought it home my mother wondered what I was doing. I quickly connected the RCA video connector to my black and white TV and booted it up for the first time. I watched everything go and for the next few months I would sit in front of that computer and learn BASIC programming. Likewise, as time would go on I would tear that small computer apart and discover a world of chips, circuit boards, and amazing technology. That large purchase would lead me to get a job at a hobby shop repairing circuit boards and building RC cars for customers. My whole life was surrounded by computers from that point forward and every waking moment was spent hunched over a computer figuring out how it worked and how I could use it to do what I needed.

Figure 2. *Commodore VIC 20*

That type of drive to learn computers is what Eben felt was missing in today's students and it drove Eben to build the Raspberry Pi. Eben wanted to see kids have a simple low cost computer they could build, use, and break. In 2009, he put together the Raspberry Pi Foundation, a charity built to promote the study of computer science in schools. The one goal of the Raspberry Pi Foundation is to help give the spirit of the hobbyist back to kids so they can create a computer from the ground up and discover the world that both Eben and I discovered as kids.

Remember the joy of opening up new computer equipment or discovering how to use a new OS? What about the first time you successfully compiled your program to do some great thing and it actually compiled without errors? Today, I am a Security Consultant and I get the opportunity to work in an environment where my hobbyist tendencies allow me to take neat tools like this and build something to make my life easier. I too have taken the Raspberry Pi and used it to create a small device I use in my own security engagements. In my Penetration Testing reports, I call it "The Raspberry Pi Test". The whole goal of this test is to see how my customer's enterprise will react to a small computer placed on their network. It's a fear all Blue Team security engineers dread and something all Red Team penetration testers should use in their bag of tricks.

It is in that spirit that I bring you this tutorial. I spent a few weeks perfecting my installations, as I am sure you will as well. But here is the basic tutorial regarding how to construct a Raspberry Pi into a penetration testing tool.

Purchasing your Raspberry Pi

In order to start this endeavor you will need to purchase a Raspberry Pi. The recommended site to purchase the Raspberry Pi is *http://www.farnell.com/pi/*. Choose your country, or if you are from the United States you can go to *http://www.newark.com/*. The country you choose will set the language, shipping and the currency option for you. Be aware that the site you choose will setup some default values and set you up for success (Figure 3).

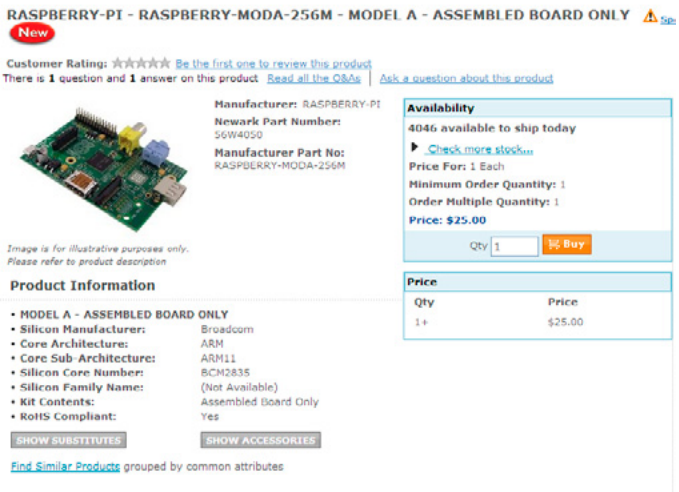


Figure 3. *Newark Website*

Assembled or Unassembled

There are many options when choosing your Raspberry Pi. You can choose to get an unassembled board or an assembled board. My soldering skills have not stood

the test of time and in so doing, I was not confident that I wanted to rely on my ability to solder the first time out of the gate. So, I purchased an assembled board. But if you are one of those people where you feel confident in your ability to solder then feel free to order an unassembled board. I have since done so and I can say the experience was great. The smell of the solder is something that sticks with you forever.

Raspberry Pi Model A or Model B

The next choice to make is what model to purchase. There are two different models called Model A or Model B. Most will want to purchase the Model B version because you will want the latest and greatest. But some on a budget may want the Model A for some sort of pet project. Model A is normally a $25 (US) investment; Model B is a $35 (US) investment. The specification differences are listed below:

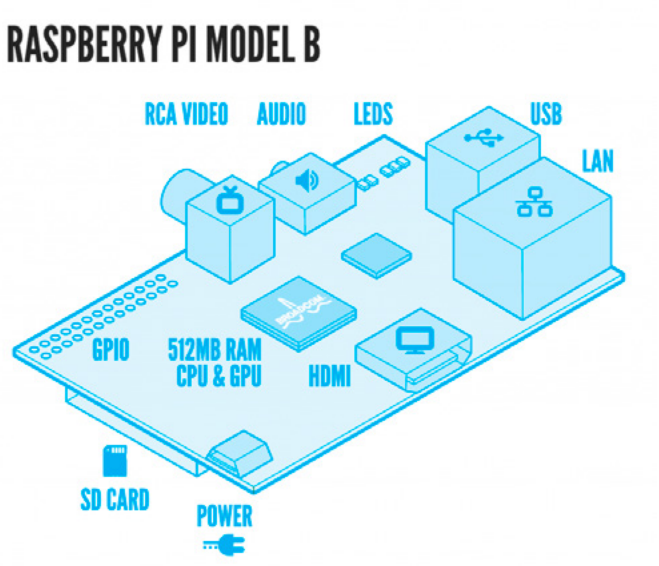


Figure 4. *Raspberry Pi Model B*

Specifications (Figure 4)

- SoC: Broadcom BCM2835 Multimedia Processor, comprised of:
- CPU: Single-Core ARM1176JZ-F (ARMv6 ISA) at 700 MHz
- GPU: Broadcom Dual-Core VideoCore IV Media Co-Processor
- RAM: 256MB (Model A & B)
- USB: 2x USB 2.0
- Video: 1x HDMI, 1x RCA Analogue Video
- Audio: 1x HDMI, 1x 3.5mm Analogue Jack
- Storage: SD Card
- Networking: None (Model A) or 10/100 Ethernet (Model B)
- Additional Connectivity: GPIO, UART, I2C, SPI, CSI, DSI, JTAG
- Actual Size: 85.6mm x 53.98mm
- Costs: Model A = $25.00; Model B = $35.00 USD

Shopping List

Of course you are going to select and purchase your Pi but, you will need a few accessories as well. Use this list to identify those items.

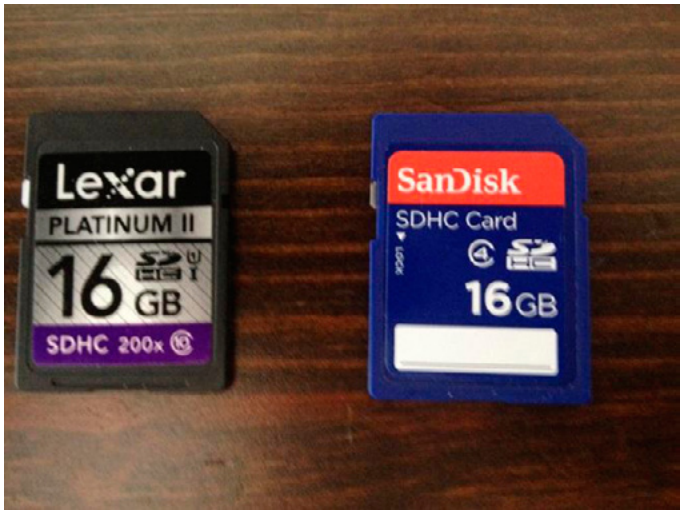


Figure 5. *Class 10 and Class 4 SD Cards*

Hard Drive

You will need to purchase a Hard Drive for your new Pi. Notice on the basic schematic there is no hard drive listed. The hard drive in the unit is the SD card so if you have one around for another project you can use it. But here is a note about the cards, it is recommended you get a card that is minimally a Class 4. I have had problems with cards under a Class 4 card. One problem I would experience is that even though I would shut down the Linux operating system correctly, the card would still have errors on it and a few times I lost the entire partition. So stick with experience and use a Class 4 or better. I am currently running a Class 10 Lexar card with 16GB of space. This is a great card and it has been rock solid (Figure 5).

Power Supply

You will need a power supply. No giant black brick will be shipped with your Raspberry Pi, you will need to purchase one or you will need to "find" one. If you are a technologist like me, you have a few power supplies lying around for the different gadgets you use. You can buy a power supply from Element 14 or you can use any power supply that

is 5V at 700mA. Many mobile phone chargers fit these criteria. I personally use my iPhone charger shown below. It makes the entire penetration testing platform nice and compact (Figure 6).

Figure 6. *Raspberry Pi and iPhone Charger*

Charging Cable

Of course, your iPhone cable is not a micro-USB power supply but, I had one of those for another accessory. So if you do not have a micro-USB supply you should get one from Element 14 (Figure 7).

Figure 7. *All Necessary Items Together*

Video

If you want to SEE your Raspberry Pi boot up you will need to plug it into an HDMI compatible resource like a TV or into a RCA video jack. I used my home TV for my testing. Again, I had spare RCA cable from an old TV project that helped me out. You may need to purchase an HDMI or RCA cable.

Raspberry Pi Case

Yes, you can purchase a case to go with your Raspberry Pi. You can make it pretty or you can make it stealth either way the cases can be found on the site, so make sure you get one that fits you. It is also a good investment because you never know where you will be placing your Pi. So, a case is a good investment to protect your new toy, which cost anywhere from $7 and up (Figure 8).

Figure 8. *Raspberry Pi Case*

Raspberry Pi Bundles

Now, if all of this is scary and you just want to click and buy a bundle, feel free to do so. Newark and others have Raspberry Pi bundles you can buy that take all the guess work out of it. In fact, they have bundles that are the complete kit including a mouse and keyboard. Because this is PenTest Magazine, I felt we would not use a keyboard and mouse. After all, we are all experienced testers who understand SSH and how to remotely connect to a Linux system. But if you want to get a complete kit to build your Raspberry Pi those are available as well.

Kits come at a cost, however. The graphic below will show you that a complete kit costs almost $85 US, whereas I spent $35 for my Pi and $7 for my case. The other items I had lying around the house being unused.

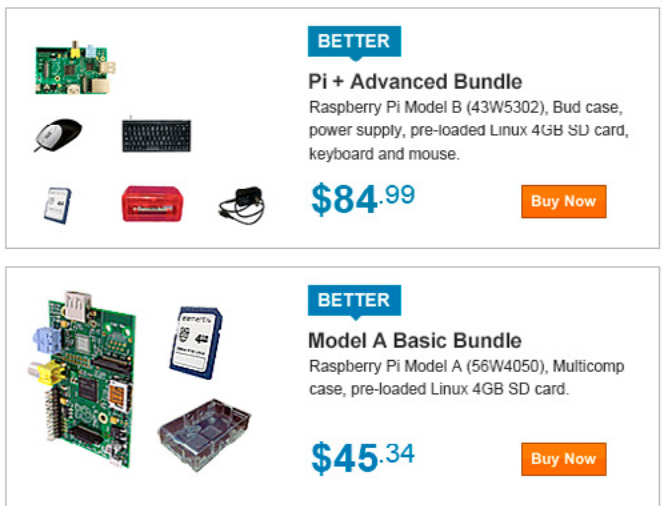


Figure 9. *Two Kits for Raspberry Pit*

Shopping Conclusion

So with those parts you are done shopping! Simply purchase and ship your new toy and feel free to unbox it with the joy you use to have during Christmas or Birthdays.

Unboxing your Raspberry Pi

Your Raspberry Pi will come in an antistatic bag with all your other goodies. As you will see, it's only a single board computer with no moving parts (Figure 10).

Figure 10. *Unboxed Raspberry Pi*

Figure 11. *Scale picture for the Raspberry Pi*

Raspberry Pi Tour

It's often hard to understand scale when you read articles. However, the Raspberry Pi is very small. I am including screenshots for readers to see and get an idea as to how tall and small the Raspberry Pi is when it arrives. As a contrast, I am using my iPhone and iPhone power supply as scale references. The iPhone used for these pictures is an iPhone 4S (Figure 11-16).

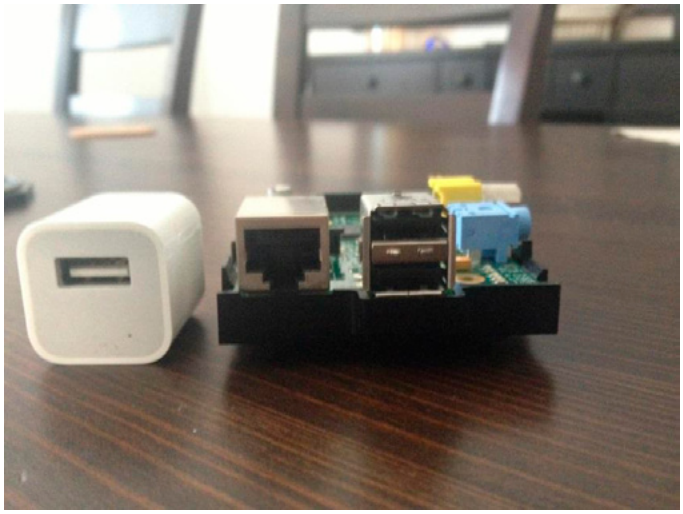


Figure 12. *Using the bottom of my Case you can see the Raspberry Pi is as tall as the iPhone charger*

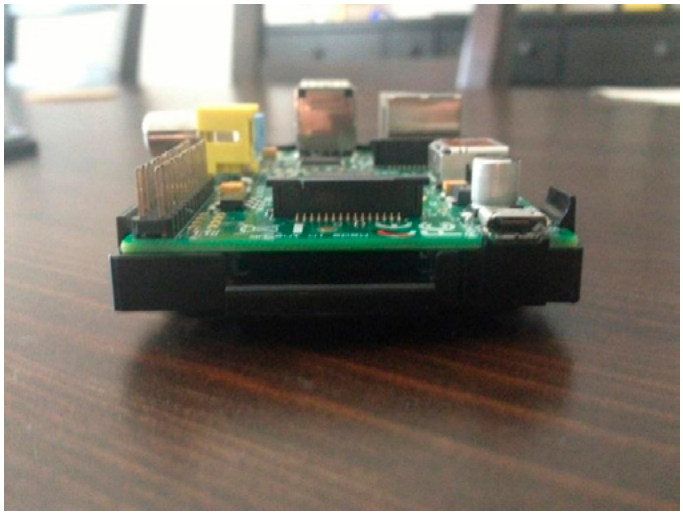


Figure 13. *SD Slot with Mini-USB power on the right*

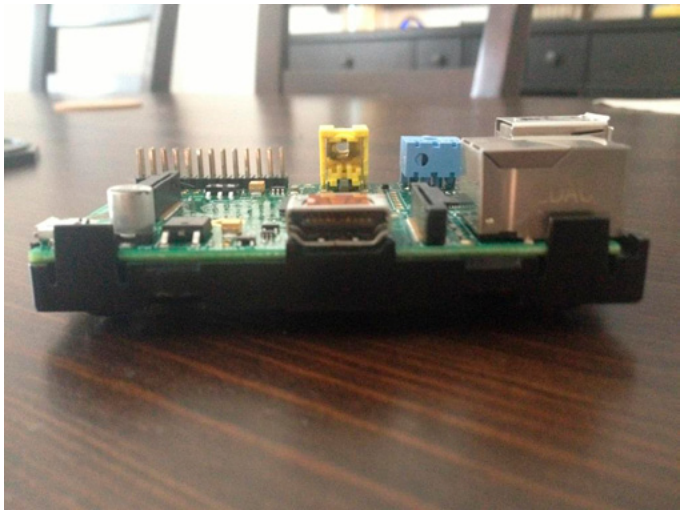


Figure 14. *HDMI side view*

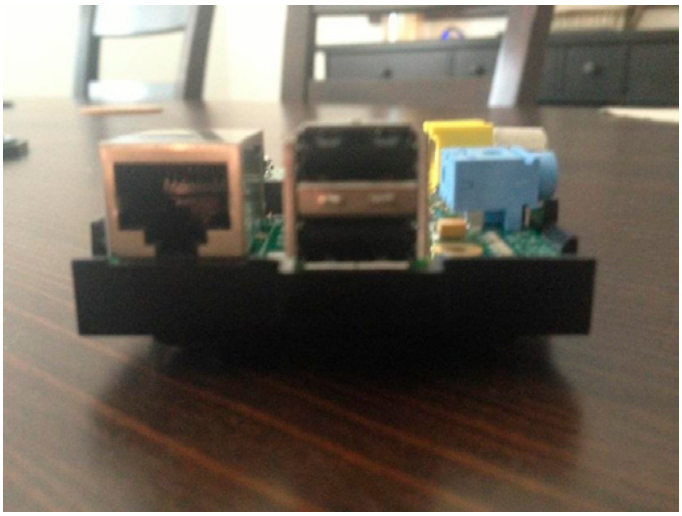


Figure 15. *RJ-45 Ethernet and two standard USB ports*

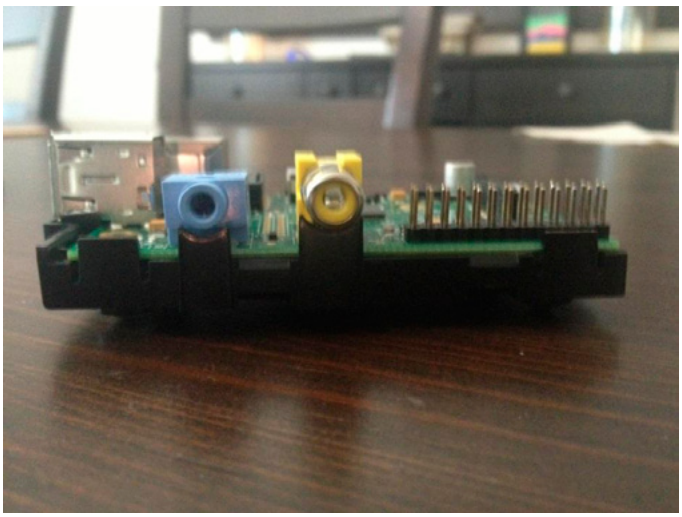


Figure 16. *Serial Audio and RCA jack with the GPIO expansion port on the right*

Walk Through Conclusion

Overall, the Raspberry Pi is a very small single board computer with more power than most of us had when we were kids. Next we will format our SD card and create a hard drive for our Raspberry Pi. Then we will load some cool tools onto the card and setup our pentesting Raspberry Pi.

Setting up our Raspberry Pi

If we plug our Raspberry Pi into its video resource and power it on, all you will get is a red light on the power. I plugged mine into RCA and power and there is no CMOS boot screen or any indication that something is happening outside of the red light. I wanted to show this to you because this is the only interface you have if something goes wrong with your Raspberry Pi or SD Card hard drive. If your partitions are damaged, or you are not giving enough power to the Pi, you will want to review these lights for an indication of what has gone wrong (Figure 17).

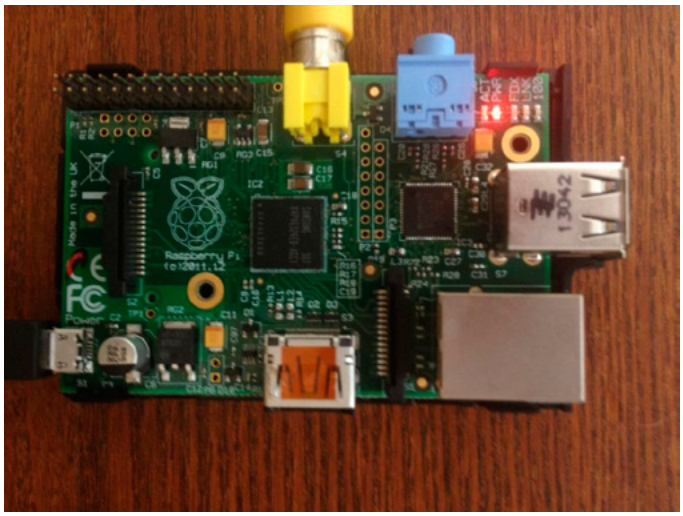


Figure 17. *Raspberry Pi Diagnostics*

Many sites document the lights on the main board and they also document the causes of each problem. I have used *http://elinux.org/R-Pi_Hub* as a troubleshooting resource and it has worked well.

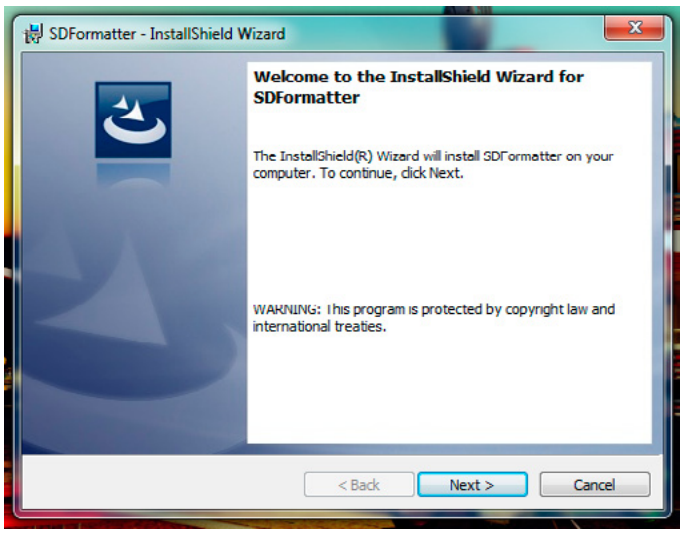


Figure 18. *SD Formatter*

Setting up the Hard Drive

The Raspberry Pi Foundation has put together a great tutorial on how to setup an SD Hard Drive for the Raspberry Pi. I will be following the guide at *http://www.raspberrypi.org* using a Windows OS in this demonstration. Obviously, if you run Linux it is easy to natively fdisk and format an SD card. The same can be said for MacOSX for that matter. However, if you want to use Windows, you want to use

an SD formatter. I have had problems using the normal format feature for a hard drive in Windows. Sometimes it just does not recognize the capacity of the entire SD Card. The Raspberry Pi Foundation mentions using this tool as well *https://www.sdcard.org/downloads/formatter_4/eula_windows/*.

Once you accept the EULA, a zip file will be sent to your system. Simply unzip and install the SETUP.EXE file and run the install. I ran the exe and clicked Next, Next, Next, Finish (Figure 18). When finished it is installed on your hard drive (Figure 19).

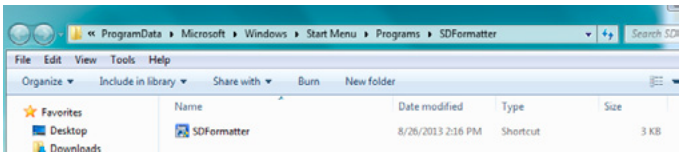


Figure 19. *Location of SD Formatter*

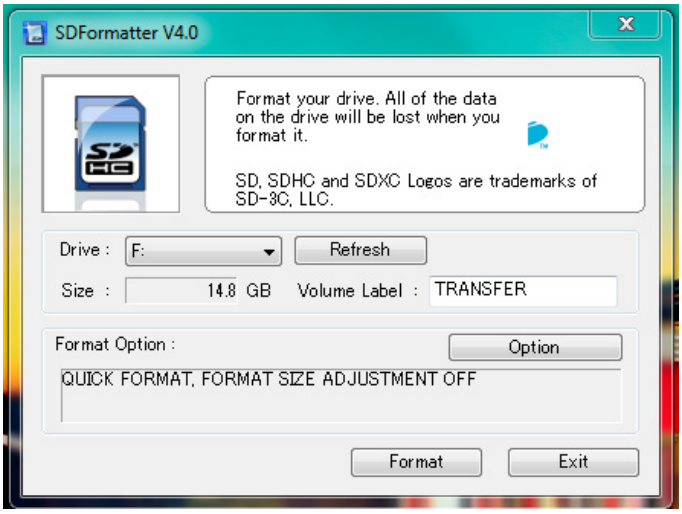


Figure 20. *SD Formatter Launched*

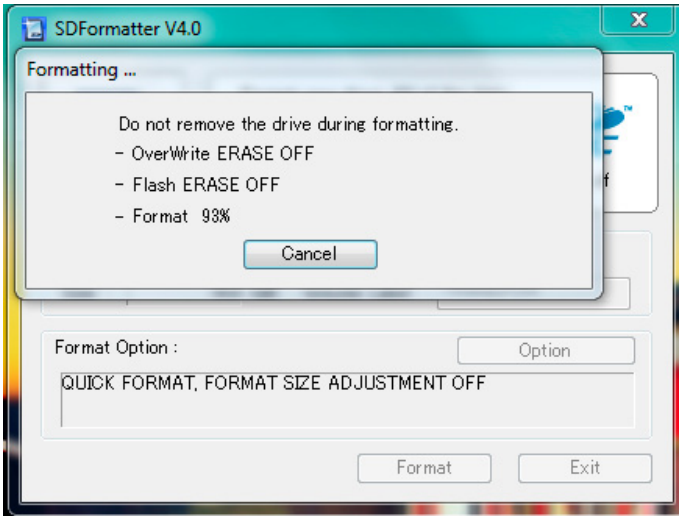


Figure 21. *Completed SD Format*

Double click the shortcut and launch the file. A simple user interface is launched (Figure 20).

You will notice in the previous graphic that my drive, size, and name of the disk were already picked up from before. You can name it anything you desire, and click format to begin erasing the drive. This will not repartition the SD Card. If you want to repartition the card you will want to use DISKPART. See the following link to partition a SD Card in Windows *http://www.winability.com/delete-protected-efi-disk-partition/* (Figure 21).

Once my format wizard is up and ready, I simply clicked Format and my SD Card was formatted and ready to go.

Prepare your Pentesting Hard Drive

Today there are a few small pentesting distributions for the Raspberry Pi. You can choose a few different flavors depending on what you want your Raspberry Pi to do. Or if you are really adventurous, you can build your own version. After all, building a pentesting system is just a matter of creating a Linux workstation and compiling some tools. But some people may like pentesting distributions because it gets you going quickly. In my review, I will talk about Linux distributions for the Raspberry Pi and show you how to install my favorite Raspberry Pi Pentesting Distro. Personally, I have a few SD cards with different distributions and "options" available. I have a special distribution that I use for WIFI cracking. I also have a special distribution for reconnaissance or "phone home" connectivity. No matter which way you want to go, you need to figure this out now so you can identify the method you will use to install an operating system.

Since my favorite distribution wants me to use the Raspberry Pi Debian version, we will move forward in that direction.

Figure 22. *Index of http://downloads.raspberrypi.org*

Linux Distributions – ARM

To start, remember that your Raspberry Pi is an ARM based computer. This means anything you use must use ARM architecture. The Raspberry Pi Foundation has put together a few different distributions ready to image at *http://downloads.raspberrypi.org* (Figure 22).

In some cases you can simply use a distribution from here. Remember, your new Raspberry Pi has some interesting connections, including that GPIO interface that will need drivers. If you do choose to build a hard drive using Red Hat Fedora, or some other Linux version, you may need to build proper drivers for your hardware. In this article, we will use the Debian version from the Raspberry Pi Foundation.

DEBIAN version for Raspberry Pi

Simply click the Debian link and choose a download type you desire. The Wheezy-armel version will work great for what we are doing (Figure 23).

Figure 23. *Choosing the download*

My personal download times run at about 7 minutes for the zip file. I never torrent for something so small, and like a good security engineer, I am going to download from a place I trust and check hashes. When the download completes, unzip your image (Figure 24).

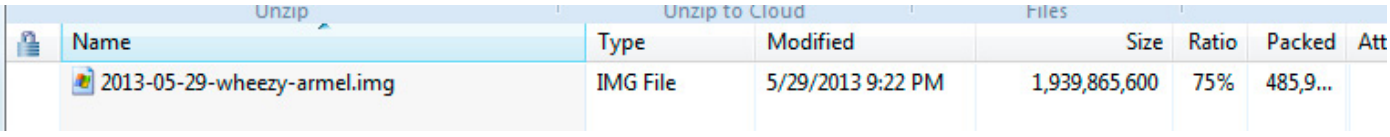


Figure 24. *Unzipper Wheezy*

Imaging your SD Hard Drive

Now that you have your Debian Image for Raspberry Pi, we can image it to your SD Card. The image will only take up 4gb of space, so I am glad I have a 16gb card. To image my SD card, I am going to use WINDISKIMAGER (Figure 25).

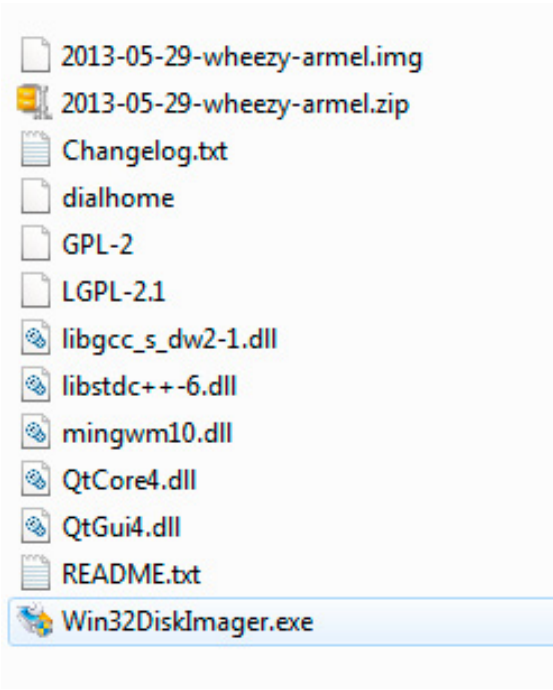


Figure 25. *Drive with Win32DiskImager and my Wheezy image*

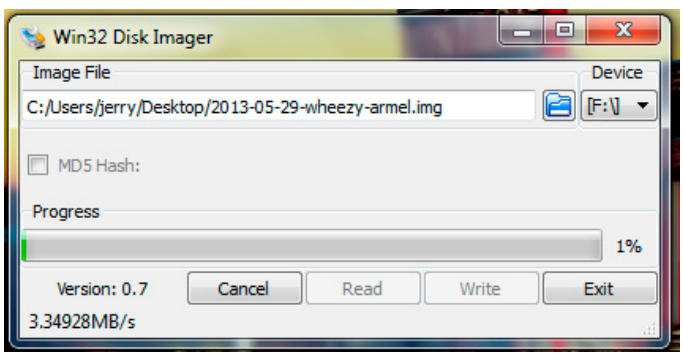


Figure 26. *Creating a SD Card Image*

Double click on Win32DiskImager if you have it, otherwise, you can get it from source forge at *http://sourceforge.net/projects/win32diskimager/*. Once it opens, select your image file using the folder icon, then check that

the image is going to the right drive, which in my case is the F drive. Once you are ready, click on the WRITE button and your SD Card will be imaged (Figure 26).

This process can take a few minutes depending on the speed of your SD Card. Here is a brief discussion about a Class 10 vs. Class 4 SD Cards. A Class 10 card can write at 10mb per second which means faster image expanding. A Class 4 card can read/write at 4mb/s. So again, a faster card could give you better results. When the write is finished you will get a "Done" message.

Image is done, now what?

Since we are using Windows, let's check out our SD Card and see what's on it (Figure 27).

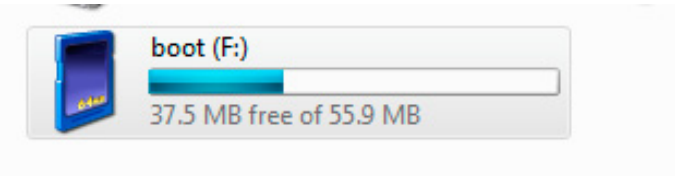


Figure 27. *Booting SD card*

Right away, you will see that the card now registers less than the full size of the SD Card. I am using a 16gb card but it reads that the F drive is 56mb and 37.5 is free. This is because the card was reformatted for the image so there are two partitions on this card. One is the Linux boot 56mb drive, and the second drive is the remainder of the 4GB image. That remainder will be your root partition once the Raspberry Pi boots up. We will expand this 4GB to my full 16GB a little later in this demo.

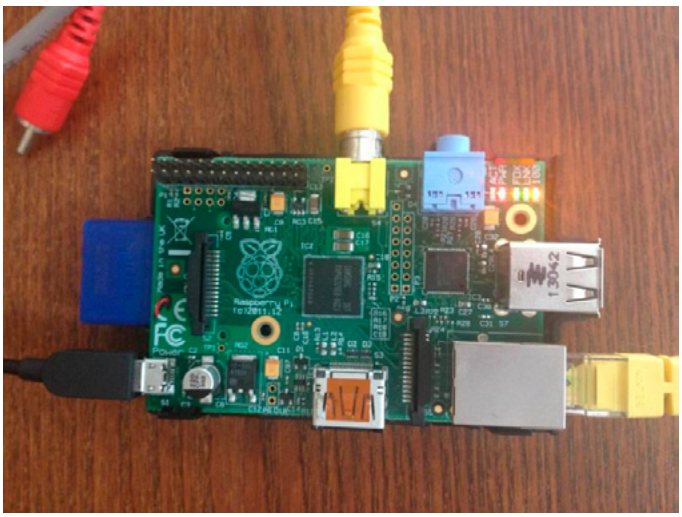


Figure 28. *First Boot*

Safely eject your card from the system and plug it into your Raspberry Pi. I am going to let it boot up and obtain an IP address on my network that is running DHCP through the Ethernet port. So that means I will need to cable up my RJ45 prior to boot.

Here you can see my Raspberry Pi ready for its first boot (Figure 28).

As you can see, my Raspberry Pi is running RCA video, RJ45 cable, power, and my SD Card is put in upside down. It only goes one way so you will figure that out. But also note that the lights on the Raspberry Pi are all lit. I have good power, it has booted, and the NIC activity lights are running. IT'S ALIVE! What do we see from my TV? (Figure 29)

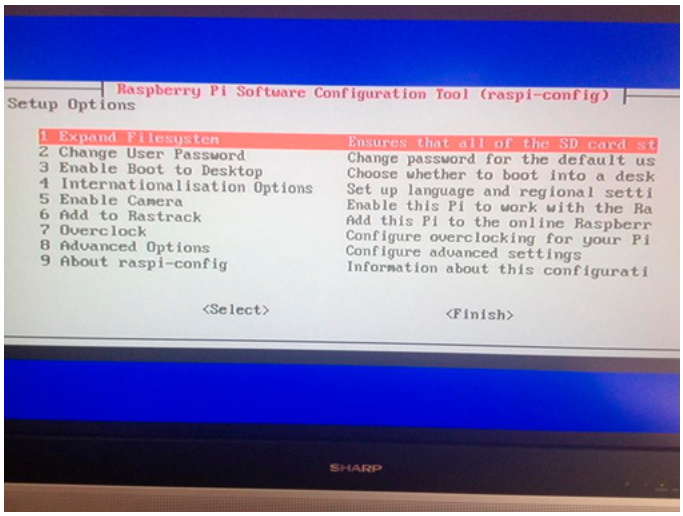


Figure 29. *Working Raspberry Pi on TV*

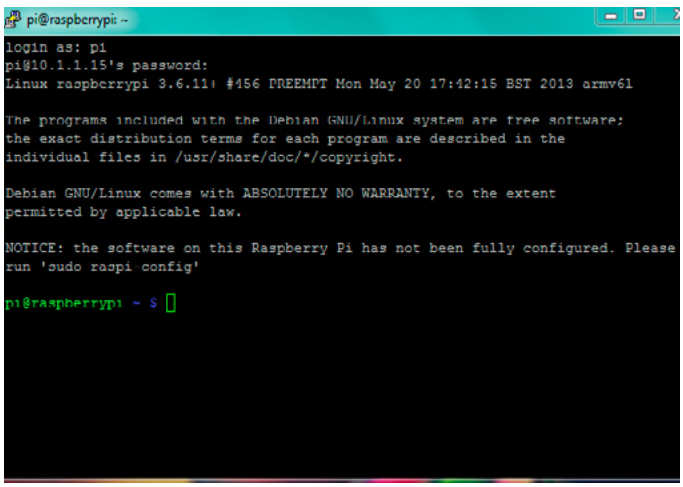


Figure 30. *First Boot Screen*

If you were to view it from the TV, you would see a normal Linux style boot up with a Raspberry Pi logo in the top left; when it's done, it goes right into the Raspberry Pi Software Configuration tool (raspi-config). This means it booted up correctly and it's ready to configure. But I don't

have a keyboard or mouse on mine. I need to SSH into my Raspberry Pi. Since I am in my office network, I can simply get the IP from my DHCP logs. If you can't identify the DHCP address, then maybe a USB keyboard is an option for you (Figure 30).

When you first SSH into your Raspberry Pi using the Wheezy image the username will be "pi" and the password is "raspberry". Take a quick look around and you will see that it's a normal Linux Debian installation. If you perform the command "df" you will see you are not using your full SD card. You need to expand the operating system to fill the full size of the SD card if you have a card larger than 2gb.

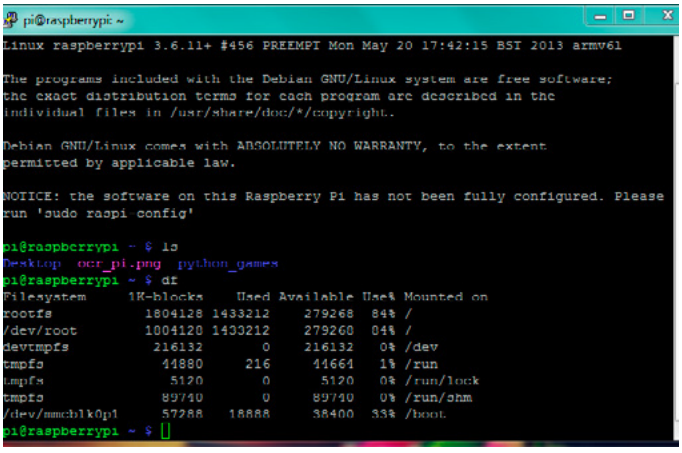


Figure 31. *Screen after using Disk Free command*

RASPI-CONFIG – Setup your Raspberry Pi

Now that you are in the console you should run "sudo raspi-config" to configure your Raspberry Pi. First we will expand the filesystem to use all the space on our SD card. Use the arrow keys in your SSH connection to select option 1 and expand the file system. When you are done, feel free to reboot your Raspberry Pi so it can finish expanding the filesystem. Here are some other features you may want to change:

- Change User Password: After all, we did just publish your username/password.
- Enable boot to desktop if you are going to use this Raspberry Pi as a desktop.
- Internationalisation Options as necessary.
- Enable Camera? Yes if you buy an Arduino connection for GPIO interface.
- Overclock – yes you can overclock your little Raspberry Pi! Use caution there is no heat sync.
- Advanced options – Check them out, easy stuff, but there is an update feature there!
- Update if you are inclined.

Normal Raspberry Pi to Penetration Testing Raspberry Pi

At this stage you have a normal Raspberry Pi using standard Linux Debian. But you don't want a regular Raspberry Pi, you want a Pi that has cool tools on it. Again, you can start here to install Header files and GCC to build your tools; or you could use a Pentest distribution. I am going to opt for a distribution so you can see how that works.

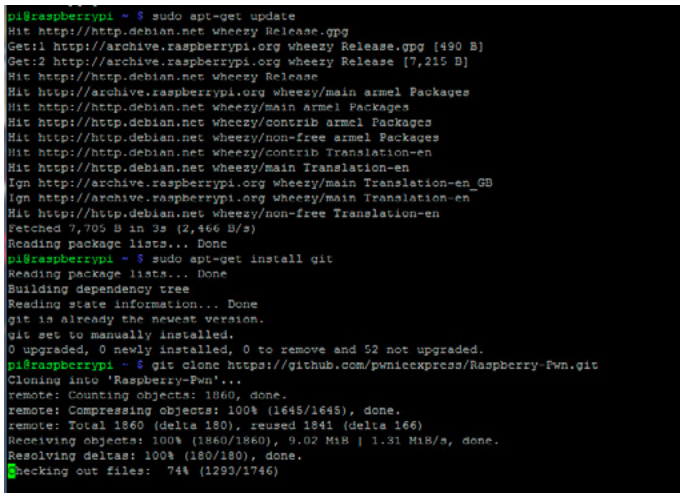


Figure 32. *APT update*

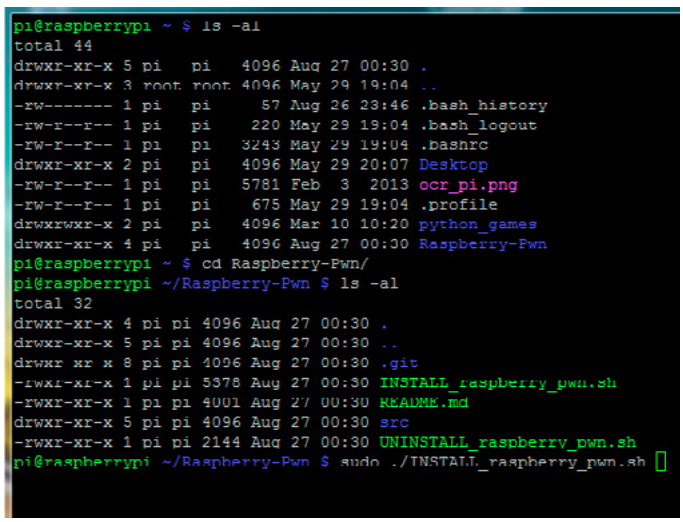


Figure 33. *PWNIE Express installation*

There are two core distributions I like for Penetration Testing. PWNPI from *http://www.pwnpi.net* has a great distribution that has some good tools. However, I really love the PWNIE Express distribution that is available in both a purchased tool and a community version. Since I see many Pentesters love Backtrack and Kali, I will opt for PWNIE Express in this demo. It's more involved to setup than others, but it does bring a bunch of great tools including SET, kismet, aircrack, netcat, and a bunch of

others ready to go. You can get PWNIE Express at *http://blog.pwnieexpress.com*.

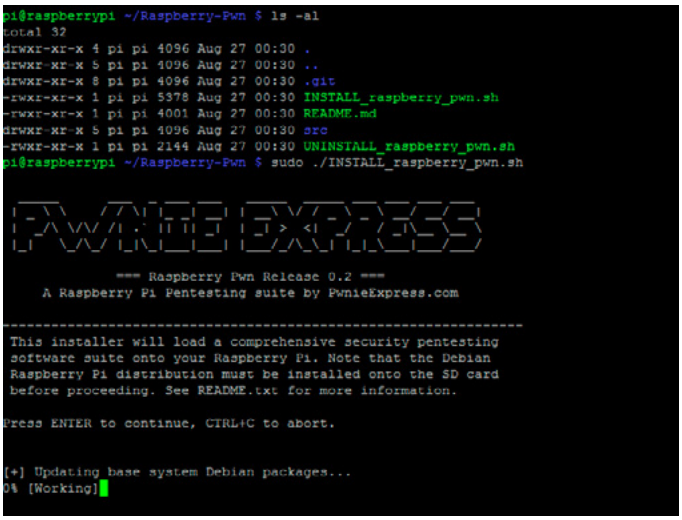


Figure 34. *GIT installation*

PWNIE Express Installation

Once you go to the blog site for PWNIE Express, you can simply follow the steps for installing GIT and running the install.

- First do the basics, ping out to confirm you have access to the internet from your Raspberry Pi and then update APT by running an "sudo apt-get update"
- After this, run "sudo apt-get install git" to install GIT.
- Finally, run the GIT command to get the PWNIE Express installer. "git clone *https://github.com/pwnieexpress/Raspberry-Pwn.git*" (Figure 32)

After the installer is installed simply run the installation command (Figure 33).

Note that you will need to change directory into Raspberry-Pwn that was created in the folder you ran the GIT command. I ran my GIT command in the home folder for

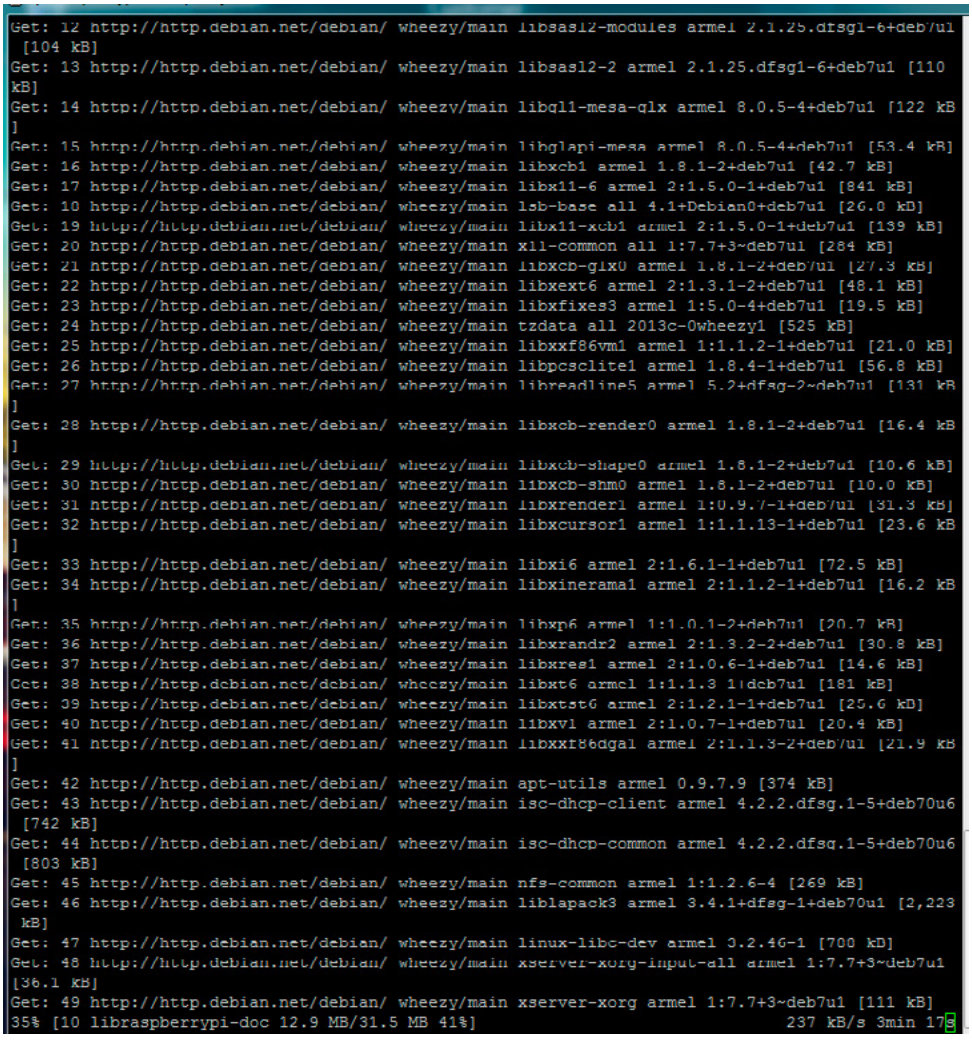


Figure 35. *PWNIE Express update*

Pi. Once you change into the Raspberry-Pwn directory, execute the command `./INSTALL_raspberry_pwn.sh`.

This command will install the GIT repository for PWNIE Express (Figure 34).

You will see the PWNIE Express installation begin updating/installing packages to support the PWNIE Express distribution (Figure 35).

This process will continue until the installation is complete. Depending on the speed of your internet connection, and class level of the SD Card, your install may take some time. My install took half an hour.

Cleaning up

At this stage of the installation you have a Raspberry Pi setup ready to perform penetration testing. Much like a BackTrack installation, many of the testing tools are placed in the /pentest folder (Figure 36). As you can see, my 16gb SD Card has 12gb of space remaining on the root partition (Figure 37). And I have a lot of free memory to use for the next engagement (Figure 38).

Overall, this new Raspberry Pi is set and ready to go. All that needs to be done is turn it on and tell it what to do.

Extending the power of the Raspberry Pi for automated attacks

In my engagements, I have programmed a script to do many things. Setup in the /etc/init.d folder, my script auto launches on boot up and performs recon scanning of an enterprise for my engagement. Then it opens up two SSH tunnels, one standard SSH reverse shell and another HTTP reverse shell. The first thing I do on an engagement is turn on my Raspberry Pi and let it work. It does a lot of the basic Recon work and simple exploitation. Fully programmable, and ready to go, a Raspberry Pi is a great tool to use on my penetration tests and, with this how-to, you can build your own in minimal time.

ABOUT THE AUTHOR

Security Consultant at Nth Generation Computing.

```
Tue Aug 27 01:31:37 UTC 2013
pi@raspberrypi ~ $ ls /pentest/
asp-auditor             dnsmap      goodfet     plecost         sqlbrute      waffit
bed                     easy-creds  goohost     revshells       sslstrip      weevely
cisco-auditing-tool     exploitdb   grabber     sickfuzz        theharvester  wifitap
cisco-global-exploiter  fasttrack   lbd         sipvicious      ua-tester     wifite
cms-explorer            fierce      metagoofil  smtp-user-enum  untidy        wifizoo
darkmysqli              fimap       miranda     snmpenum        voiper        xssfuzz
pi@raspberrypi ~ $
```

Figure 36. *Pentesting Tools*

```
pi@raspberrypi ~ $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
rootfs          15251960 2344668  12269192  17% /
/dev/root       15251960 2344668  12269192  17% /
devtmpfs          216132       0    216132   0% /dev
tmpfs              44880     208     44672   1% /run
tmpfs              5120       0      5120   0% /run/lock
tmpfs             89740       0     89740   0% /run/shm
/dev/mmcblk0p1    57288   18888     38400  33% /boot
pi@raspberrypi ~ $
```

Figure 37. *Root*

```
pi@raspberrypi ~ $ free
             total       used       free     shared    buffers     cached
Mem:        448776      55156     393620          0      10152      23780
-/+ buffers/cache:      21224     427552
Swap:       102396          0     102396
pi@raspberrypi ~ $
```

Figure 38. *Free Memory*

With the latest successful hacking attempt on the edgy Ashley Madison dating site, what are the ethical and security implications as a new thinking infiltrates the deeper and darker sides of human nature?

ROB SOMERVILLE

When the news of the Ashley Madison hack reached the public domain, there are three words that describe the emotions and mental state of a large number of their subscribers. Raw, unadulterated, fear. One member admits to being so overcome with the threat of exposure, and the corresponding shame that could entail, that he was physically sick. While much has been made over the years about the potential physical harm that technology can subject our bodies to – from repetitive strain injury and microwave radiation to poor eyesight and short attention spans amongst the social media addicted – this must be one of the first admissions in the mainstream press that the Internet can literally make you ill. Of course, it is easy to take the moral high ground and say "If you don't want the time – don't do the crime" but this ignores the inherent cognitive dissonance that goes along with all human interaction with technology. We seem to have lost that thin membrane of ethical and moral judgement that insulates us from making catastrophic decisions normally present in our day to day interactions with colleagues, friends and neighbours. To some, this is an excellent opportunity for exploitation, riches and the furthering of certain ideologies. To others, though, access via this dark portal will be costly indeed.

Unfortunately, the problem extends well past singular examples such as Ashley Madison, porn sites, drug deals on Tor, or whatever particular moral poison takes your fancy. The technology sector, like many other professional and business sectors, has swallowed whole the concept of situation ethics, where rules are based on context rather than absolutes. This is incredibly ironic, as we all know that the current generation of computers have a brutal form of logic that is simplistic in the extreme – 1 or 0, on or off, true or false. For all the abstraction, the layers of programming and intelligence, it all boils down to binary. And here lies the quandary – do we live in a universe of absolutes, good versus evil, ying versus yang, or is there a grey area in between? No matter whether the underlying architecture of technology is a true representation of moral value or not, the corresponding integration of hyper-efficiency into a society where inefficiency is de rigour spells trouble. All humans have feet of clay. Like a man walking along a cliff edge, each step is one based on faith that the ground will support his weight, yet the fool-hardy race along as if stepping on reinforced concrete.

Somewhere along the way, our institutions, our nations, our society, have turned a blind eye to the revolution that is taking place beneath our feet. We are now so much more accountable to the system, to the established order, that the slow constriction of our liberties and choices – like the frog being boiled in water – has become a regular part of life to be met with the shrug of our shoulders and a pragmatic acceptance that all will end well. In America, while there still resides a strong movement that is fiercely independent and self sufficient, the cashless, computer based society has virtually consumed society, unlike the rest of the world, where electricity and clean water could be considered a luxury. 85% of Americans are now online, and it is becoming clearer that those who are not digitally engaged will be at a major loss. Irrespective of our online status, the current mantra of efficiency, connectedness and online presence has taken root in management culture to such an extent that anyone suggesting a considered approach rather than one based on hype and stakeholder value is regarded as a heretic.

Even as far back as the 1960's, the alarm bells were ringing in popular culture as to the ramifications of computing. The Moody Blues, with the track "In The Beginning", warned us of the potential risk of becoming magnetic ink. The corresponding loss of identity, the tools of dehumanisation and calculated or perceived value under measurement (metrics) always presents a grave danger when handed to those distanced from society and real life. The psychological pathologies which drive dictatorships and fascists naturally cause them to embrace the leverage of control. And so there may be a silver lining to this incident that has morally shaken many. First of all, the hacking group may well have done the IT community a huge favour by exposing the Achilles heel of data security in terms that the general populace can relate to. In the 1800's, the Luddites were a force to be reckoned with – the British army faced down more rebellions over the mechanical loom than Napoleon's troops on mainland Europe. Hopefully, society will begin to address the cognitive dissonance that runs throughout our culture when it comes to technology, it's innovation, management and application for the greater good. Secondly, along with the other high profile attacks that have plagued the US recently, maybe the government and law enforcement will start taking the issue a bit more seriously. Assuming that 50% of the compromised records belong to US citizens, it is estimated that over 60,000 government employees will have been targeted, the same number again with top security clearance. This is a major security risk that makes the likes of the Philby and Maclean or the Profumo affairs pale into minor significance.

While the bean counters, HR drones and PR spinmeisters still have executive privilege, a comfortable window seat and the willing ear of corporate leadership, while engineers and technologists are seconded to dusty basements, out of sight, this trend will continue. Data and information security may not be at the top of the agenda quite yet, but I will be very surprised if there are not more than a few CEOs and CTOs who, after this incident, will be having a private and corporate rethink about the serious matters of risk, strategy and security.

ABOUT THE AUTHOR

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

How to Use eEye Retina On Red Hat/UNIX/Linux Systems

REBECCA WYNN

You can use eEye Retina on Red Hat/UNIX/Linux systems. In the article below, you can find some details how to make it.

What you will learn…

- How to use eEye Retina against Red Hat/UNIX/Linux systems

What you should know…

- Basic understanding of UNIX or Linux operating systems, SSH/ shell commands, and permissions.

When auditing Red Hat/UNIX/Linux systems, Retina will attempt to remotely access the target system using Secure Shell (SSH). The credential, used by Retina, must be allowed to login using SSH. The SSH server can use v1 or v2 of the SSH protocol. The authentication method must be Password based.

When configuring Retina to audit UNIX/Linux systems, a credential that is allowed to login using SSH should be added to the Retina credential manager. Usually, the credential is added as \, the typical format for win32 or win64 systems. For the UNIX/Linux systems, you do not need to add the domain part of the credential. For example:

```
Win64 Credential: MYDOMAIN\Administrator
Win32 Credential: MYDOMAIN\Administrator
UNIX credential: Administrator
Linux credential: root
```

When creating a scan job in Retina, you can select the stored credentials which allow Retina to have both a win32 credential or win64 and a UNIX/Linux credential. When the target system is scanned, the stored credentials will be tried until one is found to allow access or none are allowed.

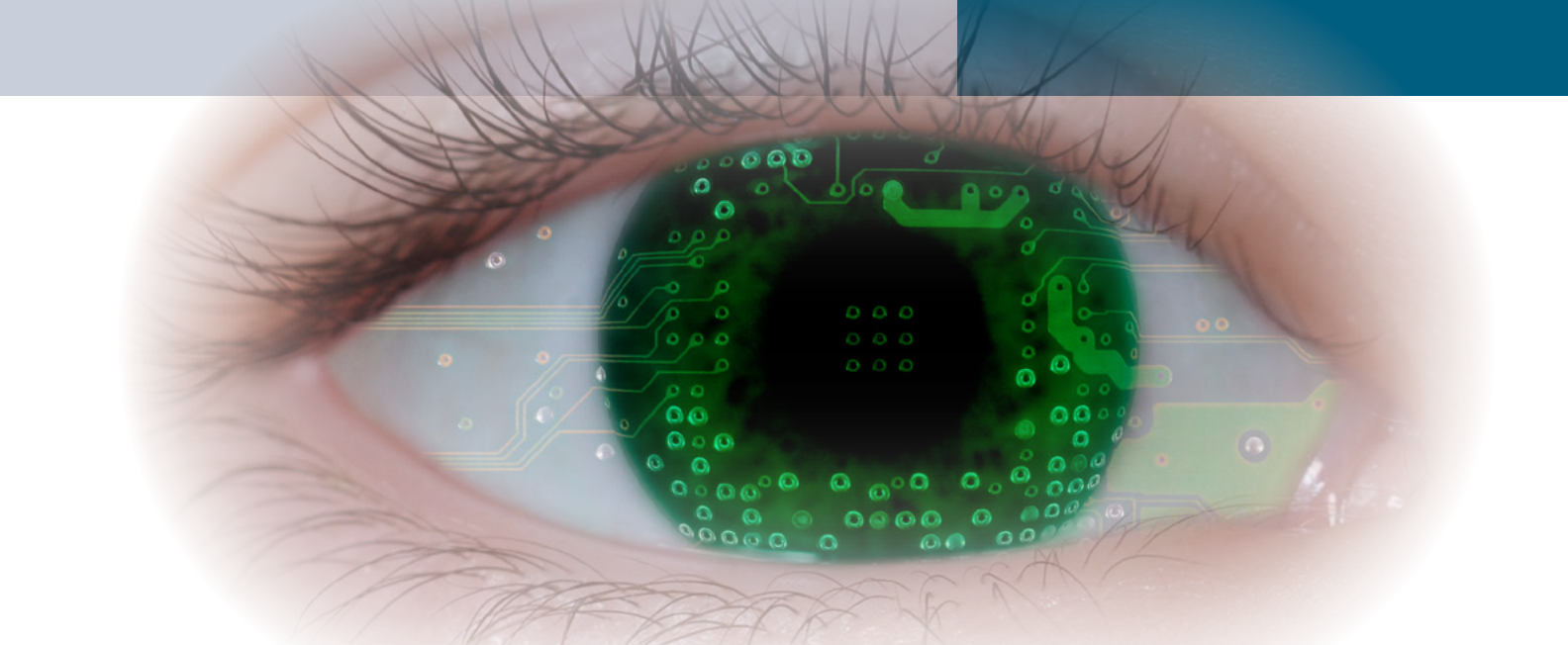
There are some configuration settings for the SSHD daemon that must be considered. Retina will only perform Password Authentication. This means the Password/Authentication option in the SSHD config file must be set to Yes.

To use the root account for access, you must also allow this in the SSHD configuration as well by setting PermitRootLogin to Yes. The Protocol can be 1 or 2 or both.

The hosts.allow and host.deny files should be configured to control access from remote systems.

eEye also recommends disabling 'Reverse DNS Lookup' configuration within SSH. This setting in SSH (on the target) can slow down Retina's scanning performance. By disabling 'Reverse DNS Lookup' on the SSH target, the target will not perform a DNS lookup after each SSH connection.

Most major UNIX/Linux vendors use a version of OpenSSH. The above referenced settings are typical of OpenSSH implementations. Specific versions of UNIX could vary to some degree. The important idea is that Retina doesn't know or have any preference to one implementation or the other. You do not need root access. It is gen-



erally a bad practice to allow root access from anywhere except the console itself. Allowing root to connect using any means remotely is not recommended. When scanning remote systems, Retina will attempt to find identifiers for known vulnerabilities through several methods. One common method is to review the package database to determine what patches could be installed. Depending on the UNIX/Linux system itself, the package database may not allow a non-privileged user access to it. If this occurs, you may need to add the user that will be used within Retina to some specific groups. SUDO support is available.

How to Enable SUDO Support for Retina

In order to provide for more flexibility for scanning of Unix/Linux targets, Retina additionally supports environments that implement the SUDO security framework. SUDO support in Retina is disabled by default and is configured through registry entries. To Enable SUDO perform the following:

1.) Use the Windows Registry Editor (Start > Run > regedit) to view the following registry key, and add the following value to this key, or modify it if the value already exists:

For 32-bit systems: `HKEY_LOCAL_MACHINE\SOFTWARE\eEye\Retina\5.0\Settings\AuditRemote`.

For 64-bit systems: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\eEye\Retina\5.0\Settings\AuditRemote`

Value: EnableSUDO
Value Type: REG_DWORD
Value Data: 0x0 (Hex) – Default (SUDO off)

2.) Set the EnableSUDO data to 1

Value: EnableSUDO
Value Type: REG_DWORD Value
Data: 0x1 (Hex) – SUDO on

Note

When scanning a UNIX system, you will want to look for this specific audit in the results to indicate if the SSH connection was NOT established during the scan. If you find this audit in the results, stop and investigate why SSH was not established and then re-scan. If you use any Audit Group other than All Audits, please ensure that this audit is included in the Audit Group before scanning.

Audit ID and Name: 2264 – SSH Local Access not available.

Additional Reference: *http://www.eeye.com/Files/Community/Retina-Best-Practices.pdf*.

Quiz Answers

1. Open Web Application Security Project
2. Do not fragment
3. No - asynchronous
4. Yes – mainly used for direct connections
5. No
6. 2007
7. Yes
8. 01110111
9. Bastard Operator from Hell
10. Yes – Stagefright
11. Henry
12. No – $92,793
13. Hans Reiser
14. 33 metres
15. MP3
16. Yes
17. IBM System/360
18. First in First Out
19. Token ring
20. Internet Engineering Task Force
21. Canada
22. William Shockley
23. Colossus
24. August 1981
25. PDP-11

ABOUT THE AUTHOR

Rebecca Wynn, DHL, MBA, CCISO, CISSP, CRISC, LPT, CWNA, CIWSA, CIWSP, MCP, MCTS SQL Server 2005, GSEC, CCSK, ITILv3, NSA/CNSS NSTISSI 4011-4016 is a Lead/ Senior Principal Security Engineer with NCI Information Systems, Inc. She has been on the Editorial Advisory Board for Hakin9 Practical Protection IT Security Magazine since 2008 and is a Privacy by Design Ambassador under Ann Cavoukian, Ph.D the Information & Privacy Commissioner for Ontario, Canada (www.privacybydesign.ca).

NET OPEN SERVICES IS AN APPLICATION HOSTING COMPANY FOCUSED ON OPEN SOURCE APPLICATIONS MANAGEMENT IN HIGH AVAILABILITY ENVIRONMENT.

NET OPEN SERVICES IS PROUD TO PROVIDE A HIGH QUALITY SERVICE TO OUR CUSTOMERS SINCE 10 YEARS.

OUR EXPERTISE INCLUDES:

- **CLOUD COMPUTING, PUBLIC, PRIVATE AND HYBRID CLOUD MANAGEMENT (OPENSTACK, CLOUDSTACK, RED HAT ENTERPRISE VIRTUALIZATION)**
- **REMOTE MONITORING AND MANAGEMENT 24/7**
- **NETWORKING AND SECURITY (OPEN BSD, IP TABLE, CHECKPOINT, CISCO,...)**
- **OS AND APPLICATION MANAGEMENT (FREE BSD, OPEN BSD, SOLARIS, UNIX, LINUX, AIX, MS WINDOWS)**
- **DATABASE MANAGEMENT (ORACLE, MYSQL, CASSANDRA, NOSQL, MS SQL, SYBASE...)**
- **MANAGED HOSTING IN CARRIER CLASS DATA CENTERS**
- **DISASTER RECOVERY**

Net Open
SERVICES

WE PROVIDE SERVICES IN EVERY STEP OF THE PROJECT LIFE, DESIGN, DEPLOYMENT, MANAGEMENT AND EVOLUTIONS. **NETOPENSERVICES** TEAM INCLUDES EXPERIENCED LEADERS AND ENGINEERS IN THE INTERNET SERVER INDUSTRY.

OUR TEAM HAS **15 YEARS OF EXPERIENCE** IN DEVELOPING INTERNET INFRASTRUCTURE-GRADE SOLUTIONS AND PROVISIONING INTERNET DATACENTERS AND GLOBAL SERVICE NETWORKS TOGETHER.

WE OFFER EXCEPTIONAL HARDWARE SUPPORT AS SOFTWARE SUPPORT ON UNIX/LINUX AND OPEN SOURCE APPLICATION. **NETOPENSERVICES** DELIVERS THESE CUSTOM-BUILT LINUX AND UNIX SERVERS, AS WELL AS PRECONFIGURED SERVERS AND SCALABLE STORAGE SOLUTIONS, TO OUR CUSTOMERS. WE ALSO OFFER CUSTOM DEVELOPMENT AND ADVANCED-LEVEL UNIX/LINUX CONSULTING SOLUTIONS.

WWW.NETOPENSERVICES.COM • CONTACT@NETOPENSERVICES.COM

Dr.WEB®
since 1992

The ERA of harmony and security

New Dr.Web! version 10

- Brand new user interface
- Configuration as simple as ABC
- Honest protection against real threats

Comprehensive protection for Windows
Anti-virus for Mac OS X and Linux

Basic protection for Windows,
Mac OS X and Linux

Protection for any and all devices*

Home and office

New purchase and renewal +150

* PC, Mac and mobile devices running OS supported by Dr.Web.

Protection for mobile devices — for free!

Doctor Web is the Russian developer of Dr.Web anti-virus software. Dr.Web anti-virus software has been developed since 1992. Doctor Web is one of the few anti-virus vendors in the world to have its own technologies to detect and cure malware. Dr.Web anti-virus software allows IT environments to effectively withstand any threats, even those not yet known.